SEGA® COMPUTER

The Official Sega User Club Magazine

JUNE ISSUE 1985

ARTICLES BY
Ian Nicholson
Colin Smith
Michael Howard

PROGRAMMES
Delta Race
Black Jack
Planet Lander
Freddie
Flexigraph
Machine Code Graphics Screen Printer/Plotter Dump
Machine Code Graphics Screen Scroll
House Hold Burglar Alarm Project and
More on Error Messages
The Sega School

INTRODUCTION

Dear Readers

As this is the last issue in the current years subscription I would like to take the opportunity to explain the obvious advantages in re-subscribing and receiving another six bi-monthly issues of your favourite micro magazine.

The Sega computer magazine aims to keep all Sega owners completely up to date with all developments concerning hardware and software. From time to time we will make special offers on Sega products through the User Club at considerable savings as a members only privilege.

We also provide you with the means to communicate with our experts when programming problems crop up in the form of our letters to the editor spot. Most important of all we provide you with program listings and information to ensure that your unit is used in all aspects of computing rather than being just another games machine.

There is of course the small matter of putting our printer and publisher out of a job and his wife, thirteen kids and two computers will be starving and destitute within a month and it will be back to the straight jackets and tranquilisers for Philip Bachler and Michael Howard. But don't let this colour your judgement, I am already confident the issue is a forgone conclusion and the re-subscriptions will come flooding in. Thank you for your support and see you next year.

Steve Kenyon

Contents

Introduction 1
Readers Letters 2
Program Dissection 3
Error Messages 6
Reviews 7
How to Program in Machine Code . . 8
Hi Resolution Scrolling 11
Disk Based Direction Program 12
Making Better Use of VRAM 15
Household Burglar Alarm 18
Computer School 21
Black Jack 22
Freddie 27
Delta Race 28
Flexigraph 30
Glossary 31

LOCAL SEGA USERS CONTACTS

AUCKLAND CENTRAL SEGA USERS CLUB
C/o 287 Broadway Furniture
Newmarket
Contact: George Shaw
Ph. 547-543

BOP USERS CLUB
Sega 102B Hinewa Road
TAURANGA
Contact: Ian Johnson
Ph. 67-349 Tauranga

CHRISTCHURCH USER'S CLUB
Contact Graham Rudman
29 Primrose Street
CHRISTCHURCH 5

GISBORNE AREA USER'S CLUB
Trevor Gardiner
Ph. 63-068 HM
or 87-175 WK

HAMILTON SEGA USER'S CLUB
P.O. Box 1548 Hamilton
Contact Leslie Wong
Ph. 384-892 Ext 66
Meetings fornightly in room KG09
University of Waikato

NAPIER SEGA USERS CLUB
Sec E.P. Lins
41 Higgins Street
NAPIER

PUKEKOHE SEGA USERS CLUB
C/o 4 Roose Avenue
Pukekohe
Contact: Selwyn
Ph. Pukekohe 86-583

ROTORUA SEGA USERS CLUB
C/- 61 Devon Street
ROTORUA

TOKOROA SEGA USERS GROUP
C/o 1 Pio Pio Place
TOKOROA
Contact Geoff Phone Number 67-105
Tokoroa

SOUTH CANTERBURY USERS CLUB
P.O. Box 73 Timaru New Zealand
President: Geoff McCayghan
Secretary: Lloyd Van Der Krogt
Contact Phs: 60-756, 61-412
TIMARU

SOUTH COROMANDEL USERS CLUB
P.O. Box 183
WHANGAMATA
Contact: Sid
Ph. 58-775 Whangamata

SOUTH TARANAKI MICROCOMPUTER SOCIETY
D.M. Beale
7A Clive Street
HAWERA

WELLINGTON USER'S CLUB
P.O. Box 187
Postal Central
Wellington
Contact: Shaun Parsons
Ph. 897-095, hm. 727-666

The above are contact names and addresses for Sega Users Clubs. If you wish to have your club advertised write to Sega Users Club P.O. Box 2353, Auckland.

If you have set up a local area user's club and you would like us to publish the details concerning your club please send them to us and we will publish the information for no charge.

READER'S LETTERS

DEAR EDITOR

Is there any way to put anything on the two borders at the top and bottom of the graphic screen on the Sega. If so, how hard is this?

A. Rowden

EDITOR'S REPLY

Unfortunately, it is not possible to put anything on these borders. You can only change their colour. There is no space in the VRAM for putting graphics in this area of the screen. If you have a look in the arcade games they do not use this area of the screen either.

DEAR EDITOR

I recently came across your magazine, 'Sega Computer'. I was very excited to find such a high quality magazine for my computer, but then to my amazement I found out it was from New Zealand, I was instantly jealous! Yousee we over the Tasman don't have anything comparable to your magazine for the Sega Computer, because of this fact I decidd to write to you and congratulate you on your magazine. I was also wondering whether any of your readers would like to exchange programs with me, I am sixteen years and am currently completing my final year at high school.

I have sent one of my programs with this letter, it is of great help to me when I am required to sketch graphs (especially when answers are not given in some text books).

Tim Werbicky
45 The Righi
Ivanhoe 3079
Victoria, AUST.

EDITOR'S REPLY

This program is on page 30

DEAR EDITOR

I've had my sega for one year and am very pleased with the software that has been produced for it, but I feel disappointed that a flight simulator hasn't come out for it. Could you tell me if a flight simulator will be made for the sega.

Michael Harris

P.S. Great magazine, keep up the good work.

EDITOR'S REPLY

Terry Johnson of Munchman Fame is working on one and it should be available in 3-4 months.

DEAR EDITOR

I think a speech synthesiser would be just great on the Sega. Do you know if there is going to be one released and, if so, could you give an approximate cost?

A. Cleland

EDITOR'S REPLY

There is an Auckland based company who have developed a speech synthesiser for the Sega. There will be more information about this in the next magazine. They have also developed a parallel interface. Please see article further on in magazine.

DEAR EDITOR

Thank you for selecting my programme for an award which I am pleased to have received. Unfortunately, it appears that the version you have published did not include the amendments which I incorporated in the later version. For inclusion in a later issue of the magazine I offer the enclosed print out of the lines which need amendment. The previous version of the Easter Calculation was found to be inaccurate.

```
1100 REM EASTER CALCULATION
1110 X=INT(Y/100)-16
1120 D=1:IF X>25 THEN D=D+1:IF X>50 TH
EN D=D+1
1130 C=3
1140 C=C+X-INT((X+D)/3)-INT(X/4)
1150 N=(Y+1)MOD19:IF N=0 THEN N=19
1160 D=(C+N*19)MOD30
1170 IF N>11ANDD>27 THEN D=D-1
1180 IF N<=11ANDD=29 THEN D=28
1190 DE=D+21:D=21:M=3:GOSUB 1000
1200 DE=DE+1:IF (DE+Z)MOD7<>1 THEN 1200
1210 D=DE:IF D<32 THEN M=3:GOTO 1230
1220 IF D>31 THEN M=4:D=D-31
1230 RETURN
4230 W=M:U=D-2:IF U<=0 THEN U=31+U:W=M-1
4450 IF Z=6 THEN Z=1:D=27
4460 PRINT "Boxing Day..";TAB(24-LEN(J
$(Z+1)));J$(Z+1);",";D+1;",";M$(12)
```

K.O. SURRIDGE

EDITOR'S REPLY

Thanks.

DEAR EDITOR

Thank you for the last copy of the magazine — it was really good. However, I seem to be having some problems with the machine code programmes. I key them all into the computer and disk drive but they refuse to work! Is there something I am doing wrong or is there something wrong with the programmes themselves?

F. Woodward

EDITOR'S REPLY

Oh, oops. What we neglected to put at the top of the machine code article's is that they are designed to work only on the memory cartridges. As the disk drive ROMS is different from that on the cartridge, you cannot get the same machine code programmes to work (this is why programmes like Munchman and the Word Procesor that run on cartridge will not run on disk drive).

PLANET LANDER PROGRAMME DISECTION

Programme by Andrew Flexman
Disection by Philip Bachler

This is a simple but challenging game which involves landing a small craft on a moving Lander Buggy.

Line 10:

This is the title. The computer ignores this line as it starts with word REM.

Line 20-70:

These lines set up the variables (Note: This can all be put on one line to make the program easier to key in and list.)

Variable A: This is the Horizontal coordinate of your space ship. (X coord.) Set to 50 Pixels across the screen from the top left hand corner.

B: This is Vertical coordinate of your ship (Y coord) It is set to 10 Pixels down the screen.

A1: This is used with the thurst to give the effect of gravity.

B1: Used for left and right movement to show momentum.

X1: This is the speed of your buggy.

Line 80:

Sends the program to line 550 where it continues working on the programme until it sees the word RETURN. When the computer spots this word it jumps back to line 90.

Line 550-700:

These are the PATTERN statements which define the sprites. (EG your ship, the Lander Buggy etc.) This information is put at the end of the program, if it is placed at the front it slows the program down.

Line 710:

RETURN so the computer jumps to line 90.

Line 90:

Switches the computer into graphics mode.

Line 100:

Colour 1 is black so this puts black writing on a black background giving a total black screen.

Line 110:

Clears the screen of any information.

Line 120:

This simply tells the Variable I to be stepped between 166 and 191.

Line 130:

The famous COLOUR statement. A Line of Random colour is draw on Lines 166 to 191 from Pixel Ø (on the very left of the screen) to Pixel 255 (on the very right of the screen).

Line 140:

Increases I by 1 and jumps the computer up to line 120 if I is not greater than 191.

Line 150:

This draws a block of colour six Pixels wide right the way across the screen. This is erased by the next 3 lines to give the top of the background.

Line 160:

Beginning of another FOR-NEXT loop. Sets I to zero and every time the computer sees the words NEXT I, I is increased by one and the computer jumps to Line 170.

Line 170-180:

These 2 lines create the surface of the planet on the "horizon." The BLINE statement removes a small randomly sized vertical section of the block drawn on Line 150. This is done 256 times right the way across the screen.

Line 190:

MAG1 is 16 × 16 Pixel sprites made up of four MAGØ 8 × 8 sprites.

Line 200-2200:

Makes stars! 5Ø Randomly coloured stars are placed in random places on the screen using PSET. RND(1) * 256 gives a Random number between Ø and 255. This is the X coordinate of th star. RND(1) * 192 gives a random number between Ø and 191. (The Y coordinate.) RND(1) * 16 gives a random number between Ø and 15. (This is the colour!) Easy when you know how isn't it?

Line 230:

Ear muff time as this line makes a noise. SOUND (Channel) 4 is one of the white noise generators your computer has built into it and 3 gives a sound like a rocket ship. 15 is the volume which is a scale of Ø-15. (15 is the loudest.)

Line 240:

This sets X (the X coord of the lander buggy) to 12Ø.

Line 250:

Generates a random direction and speed for your ship.

Line 260-270:

This checks whether the Lander buggy is too far to the left or the right. If it is the computer bounces the ship in the opposite direction.

280:

This is the beginning of the main loop that moves the ship in a random direction 20 times and then changes the direction and the speed of the lander.

Line 290:

This tests the joystick. It first checks if STICK(1) = 3 If the joystick is pressed to the right. If it is the computer adds .3 to the variable A1 (This is the momentum of the ship.) thus increasing the ships speed to the right. The tone fo the white noise is then altered by changing the pitch of SOUND (channel) 3. The computer is then sent to line 310 as if the joystick is pushed right (Which it is for the computer to be obeying this command.) it can't be pushed left so there is no need to Perform line 310. This makes the program move faster.

Line 300:

If the STICK(1) = 7 then joystick one is pressed to the left so the speed to the left is increased and the tone of the white noise changed.

Line 310:

You maybe wondering why we have 2 variables for the speed and direction of your ship. A is the horizontal position of your ship. A1 is the amount that your ship is moved left or right every time the lander buggy is moved. This means that once you start moving in a direction you continue to move in that direction until the joystick is held in the opposite direction for long enough to change the sign of A1 (If the ship is moving to the left A1 is negative and if it is going to the right A1 is positive.)

Line 320-330:

These two lines generate false gravity (Variable SPEED) and vertical momentum for your ship (This means that if you don't push the fire button the ship speeds down the screen and smashes into the ground!)

Line 340:

This puts sprite number Ø (your ship) at coordinates (A,B) with a colour of 6 (dark red.)

Line 350:

This generates the X coordinate of the Lander Buggy.

Line 360:

Puts the Lander Buggy (sprite 1) on the screen at coord (X,155) This means that it moves back and forth along line 155 down the screen.

Line 370:

The STRIG(1) statement scans the joystick 1 fire bottons. If either are pressed the momentum (B1) of your ship is lowered, the pitch of sound is changed and the shape of Sprite Ø (Your ship) is changed to make it look as though the rockets are fired.

Line 380:

To stop your ship from going of the screen this line checks its position and if any one of the coordinates are too big or too small the computer GOTO's to line 410.

Line 410:

Tests to see if your craft is low enough to touch the landing pad. If it is the computer continues at line 460.

Line 420:

Checks to stop you going off the top of the screen and changes the direction of your momentum ie you bounce off the top of the screen!

Lines 430-440:

Stops you going off the left or the right of the screen.

Line 450:

Jumps back to line 390.

Line 390:

NEXT I restarts the main loop on line 280.

Line 400:

After the Lander Buggy has moved in one direction 20 times the computer will get to this line. This sends the computer back to line 250 to generate a new random direction and speed for the LB.

Line 460:

If you are low enough on the screen (below 146) the computer tests to see if you are on the landing pad that is A and B are very close. If you have landed the computer continues to line 720.

Line 720-780:

You landed successfully (Whistles and Cheers!)

Lines 790-810:

This makes the flashing top and bottom of the screen and causes a dalay.

Line 820:

This increases the speed of the game making it harder and harder as you go.

Line 830:

Jumps back to the beginning of the game.

Line 470:

Bad news. You have not landed but you have crashed instead so there is a change in the noise.

Line 480:

This puts an explosion on the screen where you crashed your ship. Note that the coordinates have been left out.

Line 490-530:

These lines create the circles and sound that appear on the screen when you crash. The variable I is decreased from 5ØØ to 11Ø giving the sound effect.

Line 540:

Jumps to the dead routine at line 840.

Line 840:

This is it. The game is over (No tears please) these last lines just restart the program when you press the fire button.

```
10 REM Planet Lander
20 A=50
30 B=10
40 K=0
50 A1=0
60 B1=0
70 X1=0
80 GOSUB 550
90 SCREEN 2,2
100 COLOR 1,1
110 CLS
120 FOR I=166 TO 191
130 COLOR ,RND(1)*14+2,(0,I)-(255,I),RND(1)*16
140 NEXT I
150 LINE (0,164)-(255,170),1,BF
160 FOR I=1 TO 255
170 BLINE (I,170)-(I,170-RND(1)*6)
180 NEXT I
190 MAG 1
200 FOR I=1 TO 50
210 PSET (RND(1)*256,RND(1)*192),RND(1)*16
220 NEXT I
230 SOUND 4,3,10
240 X=120
250 X1=RND(1)*3+RND(1)*-3
260 IF X<50 THEN X1=4
270 IF X>200 THEN X1=-4
280 FOR I = 1 TO 20
```

```
290 IF STICK(1)=3 THEN A1=A1+.3+SPEED:SOUND 3,500:GOTO 310
300 IF STICK(1)=7 THEN A1=A1-.3-SPEED:SOUND 3,400
310 A=A+A1
320 B1=B1+.2+(SPEED/5)
330 B=B+B1
340 SPRITE 0,(A,B),0,6
350 X=X+X1+SPEED
360 SPRITE 1,(X,155),8,14
370 IF STRIG(1)=1 THEN B1=B1-1-SPEED:SOUND 3,900:SPRITE 0,,4
380 IF A<15 OR A>230 OR B<10 OR B>146 THEN 410
390 NEXT I
400 GOTO 250
410 IF B>146 THEN 460
420 IF B<10 THEN B=10:B1=-B1
430 IF A>230 THEN A=230:A1=-A1
440 IF A<15 THEN A=15:A1=-A1
450 GOTO 390
460 IF A<X+2 AND A>X-2 THEN 720
470 SOUND 4,3,15
480 SPRITE 0,,12,9
490 FOR I=500 TO 110 STEP-10
500 K=K+4
510 CIRCLE (A+8,B+20),K,RND(1)*16,,.5,0
520 SOUND 3,I
530 NEXT I
540 GOTO 840
550 PATTERN S#0,"070F1F3E7D1D1E0F"
560 PATTERN S#1,"0F07050A10206050"
570 PATTERN S#2,"E0F0F87CBEB878F0"
580 PATTERN S#3,"F0E0A0500804060A"
590 PATTERN S#4,"070F1F3E7D1D1E0F"
600 PATTERN S#5,"0F07050A15226150"
610 PATTERN S#6,"E0F0F87CBEB878F0"
620 PATTERN S#7,"F0E0A050A844860A"
630 PATTERN S#8,"00000000000000FF"
640 PATTERN S#9,"7A381A3F10285438"
650 PATTERN S#10,"00000000000000FF"
660 PATTERN S#11,"FEFCF8FC08142A1C"
670 PATTERN S#12,"0002001008093307"
680 PATTERN S#13,"0B0341001210B002"
690 PATTERN S#14,"0000942084900C4D4"
700 PATTERN S#15,"C0805004401400048"
710 RETURN
720 COLOR 0
730 CURSOR 75,80
740 PRINT "CONGRATULATIONS !"
750 CURSOR 90,90
760 PRINT "YOU MADE IT "
770 CURSOR 55,100
780 PRINT"NOW FOR YOUR NEXT MISSION"
790 FOR I=1 TO 180
800 COLOR,,,RND(1)*16
810 NEXT I
820 SPEED=SPEED+1
830 GOTO 10
840 BLINE (40,75)-(226,110),,BF
850 COLOR 0
860 CURSOR 99,80
870 PRINT "GAME OVER"
880 COLOR 0
890 CURSOR 45,95
900 PRINT "PRESS FIRE BUTTON TO TRY AGAIN"
910 COLOR,,,RND(1)*16
920 IF STRIG(1)=1 THEN SPEED=1:GOTO 10
930 GOTO 910
```

ERROR MESSAGES

In the last magazine there was a small article about Error Messages. (EM's for short.) This was obviously not enough as lots of people have contacted us asking for more. We will cover the EM's covered in last months issue in more depth as well as introducing some more of the strange little phrases which appear on screen from time to time.

SYNTAX ERRORS: This is probably the most common EM and is the easiest to fix. (thankfully!) You will get this EM when the computer has received some information that it doesn't understand. It can occur while you are keying in information or while running a programme. If you get this error while entering a command directly simply retype the line or, using the cursor arrows move up the screen and correct the problem if you can see it. Now if you still get the error and you can't see why check the following:
A. You haven't mixed up any i's, l's and one's or any o's and zero's. This is very easy to do and the computer does not understand things like:-

L1ST
SØUND

B. Make sure that there is nothing else on that line of the screen by going to the end of the line and pressing CTRL E or by clearing screen and retyping the line.
C. Reset the computer and start the line again.
D. You have got the right number of brackets and you are using the right type of bracket.

If you get a Syntax Error while a programme is running the first thing to do is to list line the concerned. Eg

? Syntax error in 100
LIST 100 [CR]

Check the line. Fix any obvious mistakes and there should be no further problem. When you list the line, if TWO lines are listed, erase the second, unwanted line, push CR, and retype the line just erased on a new empty line. You have forgotton to press the CR key at the end of the first line when entering the programme. (Naughty!)

STATEMENT AND FUNCTION PARAMETER ERRORS: These aren't so nice. You will get one of these EM's if a variable (parameter) is too big or small. Check the line over first making sure that you haven't left out any brackets (a typical cause for this EM.) or anything silly like that. If you can find no errors things get a little bit more complicated as you must work out which parameter is causing the error. To do this print all the variables and string used on that line. Eg

? Statement parameter error in 100
You must type:
LIST 100 [CR]
100 D=RND(F)*COS(B)*AD(I,K)
You must now type:
PRINT F;B;AD;I;K [CR]

If one of the numbers or letters printed is obviously wrong (ie very big or vey small) then you must look back through the programme and check every line that uses that variable. In a big programme this is not easy but it's the only way to correct the error. Other things to check for are any command that uses the graphic screen (Eg LINE, PAINT, PSET, CURSOR etc) should not have a X value between 0 and 255 and a Y value between 0 and 191. Check in the manual for definitions of the Functions and Statements used to see if there are any restrictions on the parameters which can be used. If you are still unable to fix the problem you will have to check the program line by line to find the error. (Have fun!)

OUT OF DATA ERRORS: See last issue

TYPE MISMATCH ERRORS: See last issue

UNDEFINED LINE NUMBER ERRORS: These occur when you try to send the computer to a line that doesn't exist. (Either with a GOTO, GOSUB, ON . . . ON . . . GOSUB, IF . . . THEN, and RENUM) List the line mentioned in the EM and check the GOTO's GOSUB's etc. This should fix the problem. If this EM pops up when you use a RENUM then you will have to check every GOTO, GOSUB, IF . . . THEN etc, to find the one (there might be more than one if you have not been careful.) line that sends the computer to a line that doesn't exist. This is the only way you can correct it.

DIVISION BY ZERO ERRORS: These occur when you try to divide a number or variable by zero. List the offending line and make sure it has been entered correctly. If there is no visible error list out all the variables used for division on that line and see if any of them equal zero. Check to see you have got the right amount of brackets and that they are in the right place. Also check your i's, 1's etc. If you can't find a mistake you will then have to check back through the program line-by-line looking for lines that use the variables that equalled zero in the original line.

OUT OF MEMORY ERRORS: Happen when there is no memory left for you to use. This can happen when you are loading in programs from cassette, when running programs or when typing them in. You have to make more memory available to the computer.

LOADING: There is nothing you can do but switch the computer off and try again. If you have no luck this time you are trying to load a program that is too big for the memory cartridge. (ie you are trying to fit an elephant into a Minnie.) This should only happen if you own a IIA or IIIA cartridge. There is nothing you can do about this accept buy a new cartridge.

RUNNING AND ENTERING A PROGRAM: The computer has run out of space to work on your program. There are a number of things you can do to give it more memory space. Firstly go through the program and remove all the spaces on program lines (Not between words enclosed in quotation marks in PRINT statements.) Now remove all the REM statements. Make sure that all your variables names are only up too 2 letters or characters long. (The computer only looks at the first two letters or a variable name eg. It sees LEG and LE as the same Variable.) Enter ERASE [CR] This wipes the arrays and realeases more memory. Try SAVEing the program, switching the machine off and reloading the program. If all the above listed suggestions do not fix the problem try decreasing the numbers in the DIM statements (These are normally found at the beginning of the program.)

VALUE OF SUBSCRIPT ERROR: Occurs when you use an array (EG. A(1) . . . a(101) or ab$(21) etc) List the offending line and look for the Arrays. If you have not DIMemnsioned the array with a DIM statement you will get this EM's. Check to see that there is a DIM at the start of the program. If the Array has been DIMed and you get this error, you have used a number inside the brackets on one of the arrays on the offending line, which is bigger

Cont. on p. 25

REVIEWS

Vampire Spelling

by Michael Howard

As soon as the title of this program was heard it was obvious who the programmer was, I mean who else goes ape over vampires, and Purple People Eaters? Well, this is another offering from the stable (dungeon?) of the author of "The Horse!', "Help!" and "More than 50 programs for the Sega SC2000," and features none other than Vanessa the Vampire, Fearless Fred and Slender Sue. The story-line goes something like this:-

Vanessa the Vampire got thoroughly annoyed when Throggle led the Purple People Eaters out of "The House!" and into "The Crazy Crypt," so much so, that she herself decided to leave her abode and wreak havoc on the world.

She was hanging from the rafters one night (all her coffins had been destroyed) reading a copy of "The Very vicious Vampire" (a trendy magazine for blood-suckers) when a black cat (called Saskutz) came in and said she knew of a relative of Siddy Superspook. The Vampire started licking her lips and sharpening her fangs at the thought of getting her revenge on that creep Siddy. Apparently, Siddy has a relative called Fearless Fred, and Fred is going out with a girl called Slender Sue. Vanessa, started thinking. The next day she kidnapped Sue and took her to her mate's castle, (Vanessa ended up killing her "mate" and claiming the castle for herself) and imprisoned her. Within a few minutes Fred had heard the news of his loved one's kidnapping.

The program starts when Fred has arrived at the castle and can see Sue. To rescue Sue, Fred (You!) must spell at least 13 words correctly, each time a word is spelt correctly Fred goes into his shed and builds part of a seige engine, he then walks out with a wheelbarrow and assembles the wheels, braces, ladders and gangplanks of the siege engine, when the machine is fully built, Fred can rescue Sue. Of course, things aren't quite that simple, every now and then Vanessa launches surprise attacks on Fred in her attempts to keep Sue. If you get a question wrong (You have two attempts at a word) then you are unceremoniously squashed by a 1-ton weight or a boot or a foot or get zapped by Vanessa.

The game features excellent machine code animation, over 160 sprites, title screen, music, machine code sound effects, beautiful graphics, 10 areas of spelling to choose from and wonderful sense of humour. It is easy to use, quick and for the people learning to spell . . . highly addictive and fun.

I think this program will appeal to 3 sets of people:
1. Parents wishing to teach youngsters various aspects of spelling.
2. Programmers wishing to learn aspects of animation. The animation in this program is excellent . . . the way Fred walks is really good.
3. Vanessa freaks . . . If you loose all your Freds (You have 4 lives) you get to meet Vanessa face-to-face, (NOT a pleasant experience) this is the first official close up of Vanessa.

All-in-all, no matter who buys this program, you'll love it . . . it's real fun!!!

Dungeons Beneath Cairo

WOW!! What else can I say? This game really knocked me over. This is a D & D style game with all the attributes of an arcade adventure strategic type game. I'll set the scene. You play Thor the mighty warrior, and have been sent deep beneath an pyramid to retrieve Tutankhamins Starve. Sounds easy huh? If you answer was yes then your in for a shock. The Starve is 15 levels beneath the pyramid, but to find you it must fend off wolves, dragons, mutants, ghouls, vampires, rougues, trolls and swordsman to name a few. You also have to be carefull not to fall into pits or get caught by traps. To get from one dungeon to another, you must have enough experience points. To gain experience points you can either fight monsters or sacrifice rubys at the nearest temple. Each dungeon is completely unmapped, you have to map it out. But every now and then you will come across enemies, traps or rubies.

This is not your every day game, on average it takes about two hours to complete a game. The grapics are superb and sound really amazed me. I didn't think any computer could simulate the sound of swords clanging or someone being whacked. When loading up the main program it shows a pretty picture of the pyramid. The packaging is excellent and the game is encased in a small video cassette type case. This game rivals the best of the cartridge games but is quite a bit cheaper. When it comes to games I am very fussy but this one scores 100% with me. Definately recommended. Any way I must get back to the fight, I'm being mauled by a Gargoyle.

HARDWARE REVIEW

Parallel Interface

There is now a new, exciting peripheral interface available for the Sega. This interface allows you to hook an SC3000 to any Centronics parallel printer without having to buy an SF7000 Super Control station. This means that you can get full 80 column letter quality print out from your Sega so you can use the computer for word processing, filing and spread sheets. The interface hooks up to the printer/plotter port and the power supply plugs into the interface.

A small programme is provided with the interface which once loaded in and running will allow you to use all the computers built in printer functions except you will get an 80 column print out instead of 38 columns.

This programme can be wiped from the memory and new programmes loaded in and you will still obtain an 80 column print out. All printer functions such as LLIST, LPRINT and HCOPY can be used with no alterations.

The interface has just been released in New Zealand at a cost of $200.00. For further information and ordering details please phone Auckland 403-8551. Or write to: User Tronics, 52 Stredwick Drive, Torbay, Auckland.

****

How to Program in Machine Code Language on the Sega SC3000

********** by COLIN SMITH **********

We will now start paying attention to the 'F' or flag register. The flag register is an 8 bit memory location where 6 of the bits represent the result of a logic or arithmetic operation.

7	6	5	4	3	2	1	Ø
S	Z	-	H	-	P½V	N	C

Bit Ø:-

Contains the carry flag. The carry flag is used to indicate whether the addition or subtraction just completed by the computer, generated a carry or borrow.

The carry bit can be tested and that result used for a jump, call or return in a program.

Bit 1:-

This bit is set to 1 after a subtraction and Ø after an addition. This bit is normally not used by a programmer, but is used internally by the computer when doing BCD (binary coded decimal) arithmetic. It is not possible to test this bit as it is normally for computer use only.

Bit 2:-

This bit has two functions, it is used to indicate parity and overflow If the parity is odd this bit will be set to Ø and will be set to 1 if the parity is even. The parity of data if found by adding all the '1s' in a byte, these will add to an odd or even number. It is normally used with 7 bit codes such as ASCII where the parity makes up the 8th bit.

When doing two's complement arithmetic this bit will be set to 1 if an overflow situation is detected and the sign of the number (− +) is accidentally changed.

Bit 3/5:-

These two bits are not used.

Bit 4:-

This bit is known as the half carry and is used by the computer when doing BCD arithmetic. It cannot be tested by the programmer as it is normally for the computers use only.

Bit 6:-

This bit indicates whether the result of the last operation generated a zero. In the even of the result of the last operation generated a zereo. In the event of the result =Ø, this bit is set to 1 and is reset to Ø if it is not.

Bit 7:-

This bit indicates the value of the most significant bit of the result (bit 7). If the result is positive, bit 7 will be Ø, if it is negative it will be set to 1.

8 BIT ARITHMETIC AND LOGIC

SOURCE

	REGISTER ADDRESSING							REG. INDIR.	INDEXED		IMMED.
	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)	n
'ADD'	87	80	81	82	83	84	85	86	DD 86 d	FD 86 d	C6 n
ADD w CARRY 'ADC'	8F	88	89	8A	8B	8C	8D	8E	DD 8E d	FD 8E d	CE n
SUBTRACT 'SUB'	97	90	91	92	93	94	95	96	DD 96 d	FD 96 d	D6 n
SUB w CARRY 'SBC'	9F	98	99	9A	9B	9C	9D	9E	DD 9E d	FD 9E d	DE n
'AND'	A7	A0	A1	A2	A3	A4	A5	A6	DD A6 d	FD A6 d	E6 n
'XOR'	AF	A8	A9	AA	AB	AC	AD	AE	DD AE d	FD AE d	EE n
'OR'	B7	B0	B1	B2	B3	B4	B5	B6	DD B6 d	FD B6 d	F6 n
COMPARE 'CP'	BF	B8	B9	BA	BB	BC	BD	BE	DD BE d	FD BE d	FE n
INCREMENT 'INC'	3C	04	0C	14	1C	24	2C	34	DD 34 d	FD 34 d	
DECREMENT 'DEC'	3D	05	0D	15	1D	25	2D	35	DD 35 d	FD 35 d	

ADDITION:-

Register Addressing: The operation of these codes are very simple. The contents of the source registers are added to the 'A; register and the result is placed back into the A register. e.g. The code 87 H adds the A register to itself.
The code 8Ø adds the B register to the A register.

Register Indirect: The sixteen bit code in HL is used as a memory pointer. The contents of this address are added to the A register and the result is stored in the A register.

Indexed: The sixteen bit address in the IX and IY registers are used as base values and the value 'd' is the offset value, these are then used as a memory pointer. The contents of this address are added to the A register and the result is stored in the A register.

Intermediate: The value 'n' following the opcode is added immediately to the A register and the result stored in the A register.

SUBTRACTION:-

These are the same as for addition, except the data is subtracted from the A register.

ADD/SUB with CARRY:-

These again are the same as before with the exception that any previous carry or borrow is taken into account.

AND:-

A great many people come unstuck when trying to work out logical operations. So, imagine we are down on a field full of sheep. (fig 1). The problem we have is to move the sheep to the other field.

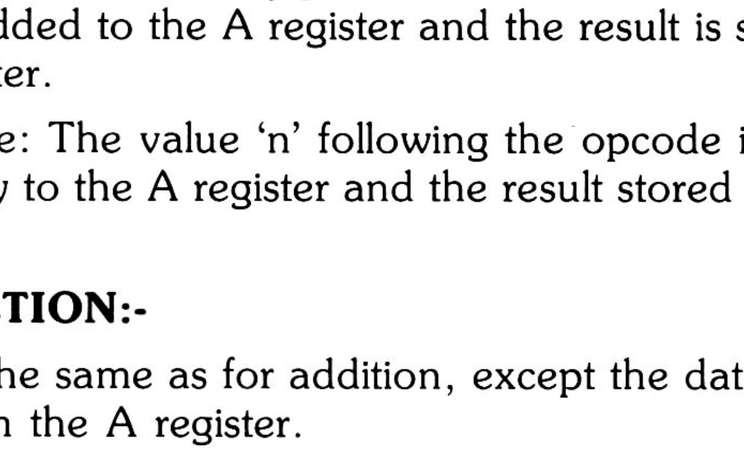


fig 1

The sheep can only be moved if gate 1 and gate 2 are open, this much should be obvious. (read this last sentence as gate 1 AND gate 2.)

If we make a gate closed equal Ø, a gate open equal 1, result not obtained equal Ø and result obtained equal 1, then we get the following Truth Table.

	Gate 1	Gate 2	Result
Gate 1 closed AND gate 2 closed	Ø	Ø	Ø
Gate 1 open AND gate 2 closed	1	Ø	Ø
Gate 1 closed AND gate 2 open	Ø	1	Ø
Gate 1 open AND gate 2 open	1	1	1

Only when both gates are open can we move the sheep.

EXAMPLE PROGRAM:-

We have the value 77_H in the A register and the value CF in the B register. Both the A and B registers are ANDed by using the opcode $A0_H$, the result is stored in the A register.

Register A = 77_H = Ø1110111
Register B = CF_H = 11ØØ1111 AND
Result = 47_H = Ø1ØØØ111

OR:-

We have the same problem as before with wanting to move the sheep, this time the gates are in a different place (fig 2).

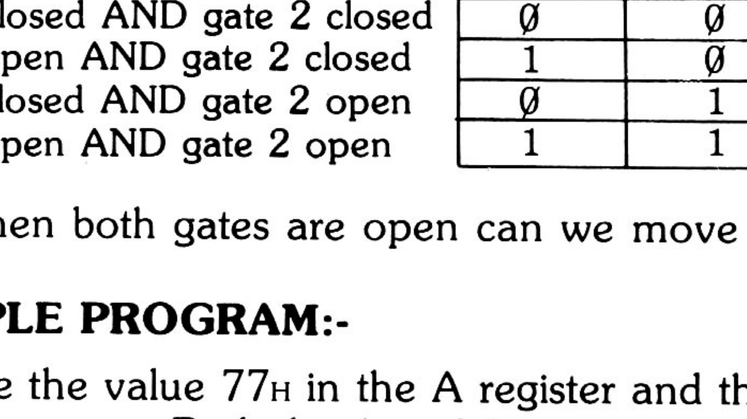


fig 2

If gate 1 or gate 2 are open, then the sheep may pass through to the next field. (read as:- gate 1 OR gate 2). Giving the gates and result the same values as for AND, we get the following Truth Table.

Gate 1	Gate 2	Result
Ø	Ø	Ø
1	Ø	1
Ø	1	1
1	1	1

Gate 1 closed OR gate 2 closed
Gate 1 open OR gate 2 closed
Gate 1 closed OR gate 2 open
Gate 1 open OR gate 2 open

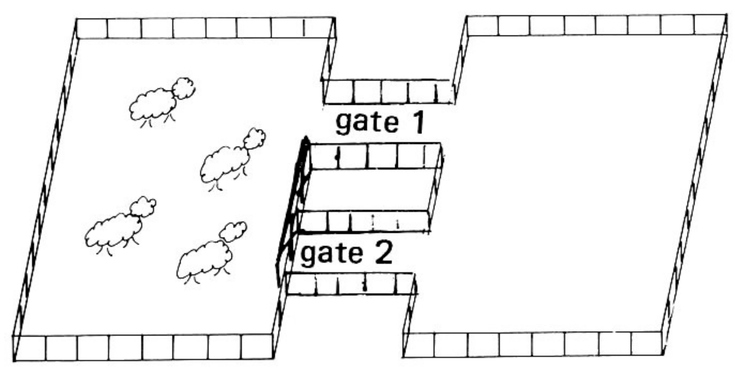
EXAMPLE PROGRAM:-

Register A = 77H and register B = CFH, using the opcode BØH will OR the two values and place the result in the A register.

Register A = 77H = Ø1110111
Register B = CFH = 11ØØ1111
Result = FFH = 11111111

XOR:- (exclusive OR)

This time we are faced with the problem that we can only open one gate at any time. The gates slide open instead of swinging open and opening one gate closes the other.



Gate 1	Gate 2	Result
Ø	Ø	Ø
1	Ø	1
Ø	1	1
1	1	Ø

Gate 1 closed XOR gate 2 closed
Gate 1 open XOR gate 2 closed
Gate 1 closed XOR gate 2 open
Gate 1 open XOR gate 2 open

The result is very similar to the OR Truth Table with the exception being that you cannot have both gates open.

EXAMPLE PROGRAM:-

Register A = 77H, register B = CFH using the opcode A8H, XORs register A and B, with the result placed back in register A.

Register A = 77H = Ø1110111
Register B = CFH = 11ØØ1111
Result = B8H = 1Ø111ØØØ

COMPARE:-

The data from the source is compared with the contents of register A. Effectivily the data is subtracted from register A, but instead of placing the result back into register A, it is deliberately forgotten. Therefore the contents of the A register are not changed. What is changed, however, is the F or flag register. This now contains the result of any special condition applying to the subtraction, e.g. zero carry etc. The flag register would normally be tested after a compare instruction being used.

INCREMENT & DECREMENT:-

These should really be self explanitory, increment adds 1 to the data referred to by the source opcode and decrement subtracts 1 from the data.

I will now describe a program, using the ADD & SUBTRACT opcodes for numbers 8 bits long. We will first do an addition using the values Ø9H and Ø8H. These must be located somewhere in memory so we will give them an address of AØØ1H and AØØ2H. After we have completed the addition we will require somewhere to store the result so we will give it the address of AØØØH.

The first step will be to load register A with the first number We will use the 8 bit Extended Addressing mode to achieve this, the opcode is 3A followed by the address which is AØØ1. We will require somewhere to store this program so we will give it a starting address of AØØ3.

AØØØH ØØH — This is where the result will be stored.
AØØ1H Ø9H — 1st byte of data.
AØØ2H Ø8H — 2nd byte of data.
AØØ3H 3AH— opcode
AØØ4H Ø1H— operand, low
AØØ5H AØH — byte first

Load A from AØØ1H.

Step 2. We will use the HL register pair to point to the second byte of data, therefore we will have to load the HL register with the address AØØ2H. Using the 16 bit load group we will use the Immediate Extended addressing mode which will load the operand into the HL register.

AØØ6H 21H — opcode
AØØ7H Ø2H — operand, low
AØØ8H AØH— byte first.

Loads HL with AØØ2H.

Step 3. We now have access to both bytes of data, so we can now add both numbers together using the Register Indirect code.

AØØ9H 86H — ADD A, HL

STEP 4. Both numbers have been summed and the result is in the A register. We now require to have the answer in A placed back into memory. We will use the 8 bit load table and find a code that loads A straight into an extended address.

AØØAH 32H — opcode
AØØBH ØØH — operand, low
AØØCH AØH — byte first.

Loads A into AØØØH.

As the computer is initially running in Basic, it will be necessary to return the machine code program back to Basic.

AØØDH C9H— Return

The Basic program to poke this program into memory and call it as a subroutine is as follows.

Remember that you are doing hexadecimal addition, if the sum adds up to greater than FF the carry would be lost. Try substituting different values for the data.

8 BIT SUBTRACTION:-

The program above will perform subtraction for us, all that is necessary is to change the addition opcode 86H to the subtraction opcode 96H. If the subtraction goes past zero the borrow will not be shown and there is no indication that the number is negative.

Next issue I will show you how to add numbers greater than 8 bits long using 8 bit addition and we shall cover 16 arithmetic opcodes.

Hi-Resolution Scrolling

by Michael Howard

Here at last, is a duo of programs specifically for movement of left or right text (ie: you can move text left or right

Enter the programs. When run, if you get an "Error" message check the DATA lines. Now save them to tape.

Okay, here is how to use them. As the programs stand they will scroll (in their respective directions) the top 8 lines of the Hi-res page. Therefore, they will scroll a sentence or word or whatever you want so long as it is only 8 pixels in height. This may seem a little restricting but let's face it, you very rarely want to scroll the whole screen. If you wish to scroll, other than the top line, this is what you do . . . Firstly select your line. Now the chosen line Must be multiplied by a multiple of eight (eg cursor 247,Ø, or 247,8 or 247,64 etc) Note the Y-direction is a multiple of 8. Now enter the following:

POKE &HFØ48, INT(Y/8)

Where Y is your Y-coordinate (which is a multiple of 8 . . . remember?!) Now place your string of info at that Y-position and execute the machine code by typing:-

CALL & HFØ58

If this all sounds a bit pukey, here's an example.

Lets say you want to scroll the word "Merlin!" From right to left do the following . . .

Enter the left scrolling program. Run it. Type in NEW (the program is still safe) and enter the following program.

```
1Ø SCREEN 2,2:CLS
2Ø A$="Merlin!"
3Ø Y=128: POKE &HFØ48, INT(Y/8)
4Ø CURSOR 2ØØ, Y:PRINT A$
5Ø FOR A=ØTO 4Ø:CALL &HFØ58:NEXT:GOTO1Ø
```

1Ø — Clear Hi-res screen
2Ø — Define a word, and place in A$
3Ø — Set Y position. Set the Y position in the machine-code. .
4Ø — Place your word at 2ØØ,Y (ie 2ØØ pixels across, Y-down)
5Ø — Call machine code 41 times. Each call scrolls 6 times. ie 6 pixels or a character.

Well . . . Thats all folks. I hope you find it useful!

Left

```
10 DATA0,3E,0,32,0,F0,D0,3C,32,0,F0,C9

20 DATAE5,CD,C5,2B,CD,C8,2B,E1,C9
30 DATAE5,CD,CB,2B,CD,CE,2B,E1,C9
40 DATAF3,6,20,C5,CD,15,F0,CB,27,F5,47
,3A,0,F0,FE,0,28,2,CB,C0,F1,CD,1,F0,78
,CD,C,F0,C1,11,8,0,AF,ED,52,10,DE,C9
50 DATA6,8,21,F8,0,C5,E5,AF,32,0,F0,CD
,1E,F0,E1,23,C1,10,F2,C9
60 DATA6,6,C5,CD,44,F0,C1,10,F9,C9
100 T=0:RESTORE:FORA=&HF000TO&HF061:RE
ADA$:B=VAL("&H"+A$):POKEA,B:T=T+B:NEXT
:IFT<>13079THENPRINT"Error"
997 REM
998 REM              Demo
999 REM
1000 A$="Hello this is a test..1234567
890-MICHAEL"
1010 SCREEN 2,2:CLS:FORA=1TOLEN(A$):CU
RSOR247,0:PRINTMID$(A$,A,1):CALL&HF058
:NEXT
1020 FORA=0TO40:CALL&HF058:NEXT
```

Right

```
10 DATA0,3E,0,32,0,F0,D0,3C,32,0,F0,C9

20 DATAE5,CD,C5,2B,CD,C8,2B,E1,C9
30 DATAE5,CD,CB,2B,CD,CE,2B,E1,C9
40 DATAF3,6,20,C5,CD,15,F0,CB,3F,F5,47
,3A,0,F0,FE,0,28,2,CB,F8,F1,CD,1,F0,78
,CD,C,F0,C1,11,8,0,AF,ED,5A,10,DE,C9
50 DATA6,8,21,0,0,C5,E5,AF,32,0,F0,CD,
1E,F0,E1,23,C1,10,F2,C9
60 DATA6,6,C5,CD,44,F0,C1,10,F9,C9
100 T=0:RESTORE:FORA=&HF000TO&HF061:RE
ADA$:B=VAL("&H"+A$):T=T+B:POKEA,B:NEXT
:IFT<>12919THENPRINT"Error"
997 REM
998 REM              Demo
999 REM
1000 A$="9876543210987654321098765432
10987654321"
1010 SCREEN 2,2:CLS:FORA=1TOLEN(A$):CU
RSOR0,0:PRINTMID$(A$,A,1):CALL&HF058:N
EXT
1020 FORA=0TO41:CALL&HF058:NEXT
```

CONTACT PAGE

Now it is our turn to whinge. Only two people have so far written in with their names and address for this contact page. There are over 2,000 members of the Users Club — this is only 0.1%!

If you want to get into contact with other Sega users in your area, or set up a club please send your name and address to us at Grandstand.

The two good souls who have sent in their names so far are:

Brain Kelly
Robert Road Otatara No 9 R.D.
Invercargill ph. 331-473

Neil Bryce
131 North street Timaru ph. 88-434

DISKED BASED ADDRESS DIRECTORY BY S.J. EASTON

This is a program specifically written for the Sega SC-3000 and SF-7000 Machines and uses both random and sequential files to fulfil its function.

Because it is written in basic the search routines located at lines 13000 and 20 20 can become quite slow as the directory nears its full point. (10-12 seconds for last entry).

A machine code routine for this task would be most helpful but to date I have not been successful in developing such a beast. (Any answers to my problem would be most welcome).

Disection of the program.
(The program is full of rem statements to make it easier to follow)

Line 10:
Erases all strings present in memory.

Line 15:
Creates DD$&DP$ (to be used later in program)

Lines 20-60:
CLS, Sets screen colour and informs user to wait while initialisation to occur.

Line 60:
Creates dimensional array for names.

Line 70:
Opens file on disk (refer page 123 of manual).

Line 80:
Length of file and if the length is "Ø" (i.e. file does not exist the routine at line 2900 is called.

Line 2910:
Creates random file on disk (note you must have a disk with at least 90k bytes free set in drive or an error will occur and the program will crash).

This instruction tells the disk drive to reserve 300 spaces on the disk for information storage each space or sector is 256 bytes long, therefore nearly 19 tracks (76800 bytes actually) of the disk are used.

More effecient use of the disk space can be achieved by making this file half this size and putting twice as much information in each sector. As you can see I've put 90 bytes on the each sector.

Some tricky program modifications are required but I did it — so you'll manage too.

Line 2930:
Fills dim created in line 60 with spaces.

Line 2940:
Calls sub routine to save the sequential file on disk. The sequential consists of all the information stored in the N$ array. (This time they are full of spaces so that as the directory gets fuller no more disks space is required). After finishing this sub routine at line 2450 the program jumps back to the main route at line 130.

These lines close "A direct DTA" file. This is the main file holding all addresses and telephone numbers "B direct DTA" is opened. This is the sequential file that holds the name info, and reads it into N$ array.

Line 130:
Sets sequential file update marker to 0. This marker changes if any addition or deletion is made and subsequently a re-write of the file on disk is made when the directory is closed.

Lines 140-240:
Displays menu page, prompts for an entry of choice and in line 240 directs program branching as indicated.

Line 1010: Option 1: read from directory.
Sets title page of option

Line 1020:
Directs program to name search sub routine.

Line 1310:
Prompts for input of name and upon input asks you to wait while search is made.

Lines 1320-1350:
Performs search, allows premature escape from search loop upon finding name and returns you to main program.

Line 1040:
Directs program to file retrieval sub routine (from disk) this routine retrieves address and phone number information from disk, prepares and displays it.

Line 1050:
Gosubs to display next option choice (another search or back to main menu)

Lines 1070-1090:
Executes choice. If the name that was entered on line 1310 was not found in the directory by the search in line 1030 the computer tells you and the option choice routine is called.

Lines 2000-2580: Option 2: addition to directory.
The program searches for free space in the directory (2039-2050). If no free space is found Line 2060 tells you and after a delay returns you to the main menu upon finding free space, F is noted (F is the file number) prompting and entering of information occurs on Lines 2080-2120 the sub routines at 2500, 2540 and 2570 ensure specified lengths of this information. This was necessary so that on retrieval of data in option it was simpler to display files.

Lines 2130-2160:
These prepare the information for storeage on disk in file number "E;; in the random file "a direct. DTA" sfud is changed from 0 to 7772 (don't ask why) to indicate that you need to re-write the sequential file when the directory is closed. (Don't ask why!)

Lines 3000-3190: Option 3: delete from directory.
The same search as used in option 1 is used to find the name you wish to delete. Once found a prompt is given to ask if the name is to be deleted. If answered [Y] the appropriate files are filled with spaces from DD$. If answered [N] then the choice of another name or the main menu is offered.

Lines 4000-4110: Option 4: close directory
This allows all 3000 names (maximum) to be listed and viewed. This is achieved in 10 blocks of 30 names and a return to the main menucan also be achieved after each block.

Lines 5000-5140: Option 5: close directory.
Line 5020 determines whether or not a re-write of the sequential file is necessary. If SFUD = 7772 then the program simply indicates that the directory is closed and halts (Line 5040) if an addition or deletion has occurred then SFUD=7772 (Line 2160 or 3140) and a re-write of the sequential file "direct B.DTA" is carried out before the prog rams halts. (Lines 5030&5100-5140)

```
10 ERASE
15 DD$=" DP$="    "
20 CLS:COLOR1,7
30 CURSOR 5,6:PRINT "DISK BASED ADDRESS DIRECTORY"
40 CURSOR 8,8:PRINT "By S.J.Easton (C) 1985"
```

```
50 CURSOR 1,10:PRINT "PLEASE WAIT FOR FILE INITIALISATION"
60 DIM N$(300)
70 OPEN "A direct.DTA"AS #1
80 IF LOF(1)=0 THEN GOTO 2900
90 CLOSE
100 OPEN "B direct.DTA"FOR INPUT AS#1
110 FOR N=1 TO 300:INPUT #1,N$(N):NEXT N
120 CLOSE
130 SFUD=0
140 BEEP:CLS:CURSOR8,1:PRINT"** ADDRESS DIRECTORY **"
150 CURSOR13,4:PRINT"**   MENU   **"
160 CURSOR13,5:PRINT""
170 CURSOR5,7:PRINT"1  READ FROM DIRECTORY"
180 CURSOR5,9:PRINT"2  ADDITION TO DIRECTORY"
190 CURSOR5,11:PRINT"3  DELETE FROM DIRECTORY"
200 CURSOR5,13:PRINT"4  LIST NAMES IN DIRECTORY"
210 CURSOR5,15:PRINT"5  CLOSE DIRECTORY"
220 CURSOR8,19:PRINT "SELECT MENU (1) THRU (5) "
230 A$=INKEY$:IFA$<"1"ORA$>"5"THEN 230
240 ONVAL(A$)GOTO  1000,2000,3000,4000,5000
1000 REM #### read from directory ####
1010 BEEP:CLS:FOR X=1 TO 100:NEXT :CURSOR 5,1:PRINT "** READ FROM DIRECTORY **"
1020 GOSUB 1300
1030 IF F=0 THEN PRINT :PRINT "NAME NOT ON FILE":GOTO 1050
1040 GOSUB 1500
1050 GOSUB 1400
1060 J$=INKEY$
1070 IFJ$=CHR$(13)THEN1000
1080 IFJ$=CHR$(77)THEN 140
1090 GOTO 1060
1300 REM ###### search for name ######
1310 CURSOR0,5:INPUT "PLEASE ENTER SURNAME:";K$:F=0:PRINT :PRINT :PRINT"PLEASE W
AIT   SEARCHING DIRECTORY":PRINT :PRINT
1320 FORN=1TO 300
1330 IFK$=LEFT$(N$(N),LEN(K$))THEN F=N:N=301
1340 NEXT
1350 RETURN
1400 REM ######### menu option #########
1410 CURSOR 4,22:PRINT "[CR]=NEW NAME        [M]=MENU"
1420 RETURN
1500 REM ### get plus display file ###
1510 OPEN "A direct.DTA"AS #1
1520 GET#1,F;AD$,0,70;PH$,70,90
1530 CLOSE
1540 AF$=RIGHT$(AD$,20):AE$=MID$(AD$,31,20):AD$=LEFT$(AD$,30)
1550 PRINT N$(F)
1560 IF AD$="  " THEN 1580
1570 PRINT AD$
1580 IF AE$="  " THEN 1600
1590 PRINT AE$
1600 IF AF$="  " THEN 1620
1610 PRINT AF$
1620 PRINT "Phone ";PH$
1630 RETURN
2000 REM ### addition to directory ###
2010 BEEP:CLS:FOR X=1 TO 100:NEXT :CURSOR 5,1:PRINT "** ADDITION TO DIRECTORY ** "
2020 REM ### search loaded sequential                    file for free space ###
2030 PRINT :PRINT :PRINT "PLEASE WAIT   SEARCHING FOR FREE SPACE"
2040 FOR N=1 TO 300:IF N$(N)="                    "OR N$(N)=""  THEN F=N :N=305
2050 NEXT N
2060 IF N=301 THEN CURSOR 1,8:PRINT "FILE FULL     ADDITION NOT POSSIBLE":FOR D=
1 TO 750:NEXT :GOTO 140
2070 REM ### layout for data entry ###
2080 CURSOR 1,8:INPUT "NAME          ";N$(F):N$(F)=LEFT$(N$(F),18)
2090 CURSOR 1,10:INPUT "STREET        ";AD$:AD$=LEFT$(AD$,30):GOSUB 2500
2100 CURSOR 1,12:INPUT "SUBURB        ";AE$:AE$=LEFT$(AE$,20):GOSUB 2540
2110 CURSOR 1,14:INPUT "CITY          ";AF$:AF$=LEFT$(AF$,20):GOSUB 2570
2120 CURSOR 1,16:INPUT "PHONE         ";PH$
2130 AD$=LEFT$(AD$,30)+LEFT$(AE$,20)+LEFT$(AF$,20)
2140 OPEN "A direct.DTA"AS #1
```

```
2150 PUT#1,F;AD$,0,70;PH$,70,90
2160 CLOSE:SFUD=7772
2170 GOSUB 1400
2180 J$=INKEY$
2190 IFJ$=CHR$(13)THEN2000
2200 IFJ$=CHR$(77)THEN140
2210 GOTO 2180
2500 REM ######## string packing #######
2510 AD=30-LEN(AD$)
2520 AD$=AD$+LEFT$(DD$,AD):RETURN
2540 AE=20-LEN(AE$)
2550 AE$=AE$+LEFT$(DD$,AE):RETURN
2570 AF=20-LEN(AF$)
2580 AF$=AF$+LEFT$(DD$,AF):RETURN
2600 STOP
2900 REM ### open main random file on disk for first time ###
2910 PUT#1,300;AD$,0,70;PH$,70,90:CLOSE
2920 REM  open memory for sequential
2930 FOR N=1 TO 300:N$(N)="  ":NEXT N
2940 GOSUB 5110
2950 GOTO 130
3000 REM ### delete from directory ###
3010 BEEP:CLS:FOR X=1 TO 100:NEXT :CURSOR 5,1:PRINT "** DELETE FROM DIRECTORY ** "
3020 GOSUB 1300
3030 IF F=0 THEN PRINT:PRINT  "NAME NOT ON FILE":GOTO 3150
3040 GOSUB 1500
3050 CURSOR 5,19:PRINT "[Y]=DELETE        [N]=REMAIN"
3060 J$=INKEY$
3070 IF J$="Y" THEN 3100
3080 IF J$="N" THEN BEEP:GOTO 3150
3090 GOTO 3060
3100 BEEP
3110 OPEN "A direct.DTA"AS #1
3120 PUT#1,F;DD$,0,70;DP$,70,90
3130 CLOSE
3140 N$(F)=DP$:SFUD=7772
3150 GOSUB 1400
3160 J$=INKEY$
3170 IFJ$=CHR$(13)THEN3000
3180 IFJ$=CHR$(77)THEN 140
3190 GOTO 3160
4000 REM #### List names on file #####
4010 A=1:B=30:C=1
4020 BEEP:CLS:FOR X=1 TO 100:NEXT :CURSOR 4,1:PRINT "** LIST NAMES IN DIRECTORY
**":PRINT :PRINT " Page ";C;" of 10":PRINT
4030 FOR X=A TO B
4040 PRINT N$(X),:NEXT
4050 A=A+30:B=B+30:C=C+1
4060 CURSOR 4,22:PRINT "[CR]=CONTINUE       [M]=MENU"
4070 J$=INKEY$
4080 IF J$=CHR$(13) AND B=330 THEN 4010
4090 IF J$=CHR$(13) THEN 4020
4100 IF J$=CHR$(77) THEN 140
4110 GOTO 4070
5000 REM ######### Close File #########
5010 BEEP:CLS:FOR X=1 TO 100:NEXT :CURSOR 8,1:PRINT "** CLOSE DIRECTORY **"
5020 IF SFUD<>7772  THEN 5040
5030 CURSOR 10,6:PRINT "CLOSING DIRECTORY":CURSOR 1,10:PRINT "PLEASE WAIT WHILE
DISK IS UPDATED":GOSUB 5050
5040 CURSOR 11,15:PRINT "DIRECTORY CLOSED":END
5050 KILL "B direct.DTA"
5060 GOSUB 5110
5100 REM ### Sequential file update ##
5110 OPEN "B direct.DTA"FOR OUTPUT AS#1
5120 FOR N=1 TO 300:PRINT #1,N$(N):NEXT N
5130 CLOSE
5140 RETURN
```

MAKING BETTER USE OF THE SEGAS VIDEO RAM

BY IAN NICHOLOSON

For those of you who are writing programs using the video ram may have wondered if there is an easier method to draw and save pictures rather than entering lots of commands, which take time to execute everytime a new picture is drawn.

A utility known as "GRAPHIC DESIGNER" (which some of you may already have) has been developed which allows you to draw your own masterpieces, and then save the resulting picture on tape. The program has been developed such that the memory space needed to store a picture is proportional to the detail of the picture: eg. a simple picture would typically require less than 1k of memory space. Thus it is possible to store more than one picture in a program. Further, almost instantaneous changes are possible between pictures because of the machine code utility which has been developed. (explained later).

The picture information is stored in free memory area above that used by your basic program. So, depending on the length of your basic program, the complexity of the pictures, and the size of memory cartridge, determines the number of pictures which can be stored.

The GEOGRAPHY programs have been also developed using the above technique.

The question that needs to be addressed is that now you have used the GRAPHIC DESIGNER program to develop your picture and it is saved on tape then how does one make use of this picture for their own programs. To explain the proceedure it is necessary to explain how the picture is stored.

THE SEGA VIDEO RAM.

The following diagram comes from page 148 of your SEGA user manual.

&HOOOO	Graphic 2 mode Pattern generator table (6144 bytes)
&H1800	
&H200	Graphic 2 mode colour table (6144 bytes)
&H3800	Graphic 2 mode pattern name table
&H3B00	Sprite attribute table
	Empty
&H3C00	Text mode pattern name table
	Empty

Areas of interest are those from &HØØØØ to &H1800 (picture detail) and &H2000 to &H3800 (colour information). The picture that is seen in SCREEN 2,2 mode is stored in these locations. By VPEEKing these memory locations it is possible to visualise the makeup of a picture's elements and colours.

So that a picture can be stored on tape it is necessary to take the information from the video RAM to the normal user RAM area. However it is not necessary to transfer all the data from the video RAM to the user RAM. All that is required is that which is necessary to recreate the picture. For example, assume that the picture is very simple consisting of a straight line. The information needed to recreate the picture is the lines location, pixel and colour.

A machine code routine has been developed within the GRAPHICS DESIGNER program which scans the video RAM looking for information which was different from the previous sample. If the information is different then the information plus its address is stored in the user RAM. Conversly, if two consequtive samples have the same information then nothing is stored in the user RAM. This in effect means the amount of memory used is proportional to the compexity of the picture.

GRAPHIC DESIGNER PROGRAM

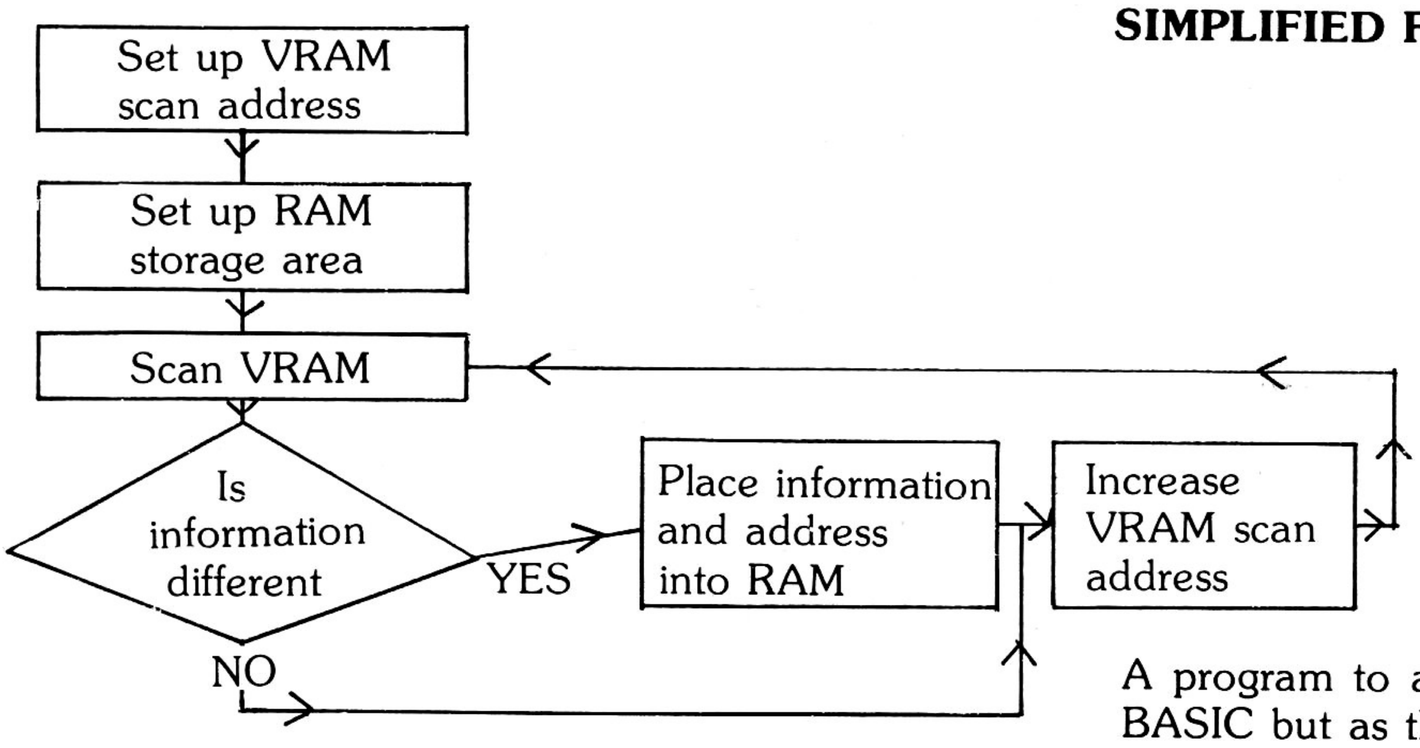
The program within the GRAPHIC DESIGNER does not make full use of the video memory as the top part of the screen is used for colour and function selection whereas the lower part is used for instruction messages. This means that instead of using &H0000 to &H1800 of the VIDEO RAM for pixel information, the area &H0100 to &H1600 is used. Similary the area &H2100 to &H3600 is used for the colour information. This is reflected in the diagram shown below.

&H0000	Colour and function selection
&H0100	Picture pixel information
&H1600	Instruction area
&H1800	
&H2000	Colour and function selection
&H2100	Picture colour information
&H3600	Instruction area
&H3800	

MACHINE CODE ROUTINE FOR STORING PICTURE INTO RAM

The routine shown in the flow diagram below scans the video ram from &H0100 to &H1600 storing the necessary information into the user RAM beginning at address &HB000. It then scans the VRAM from &H2100 to &H3600 storing the required information above that already stored above &HB000.

SIMPLIFIED FLOW DIAGRAM



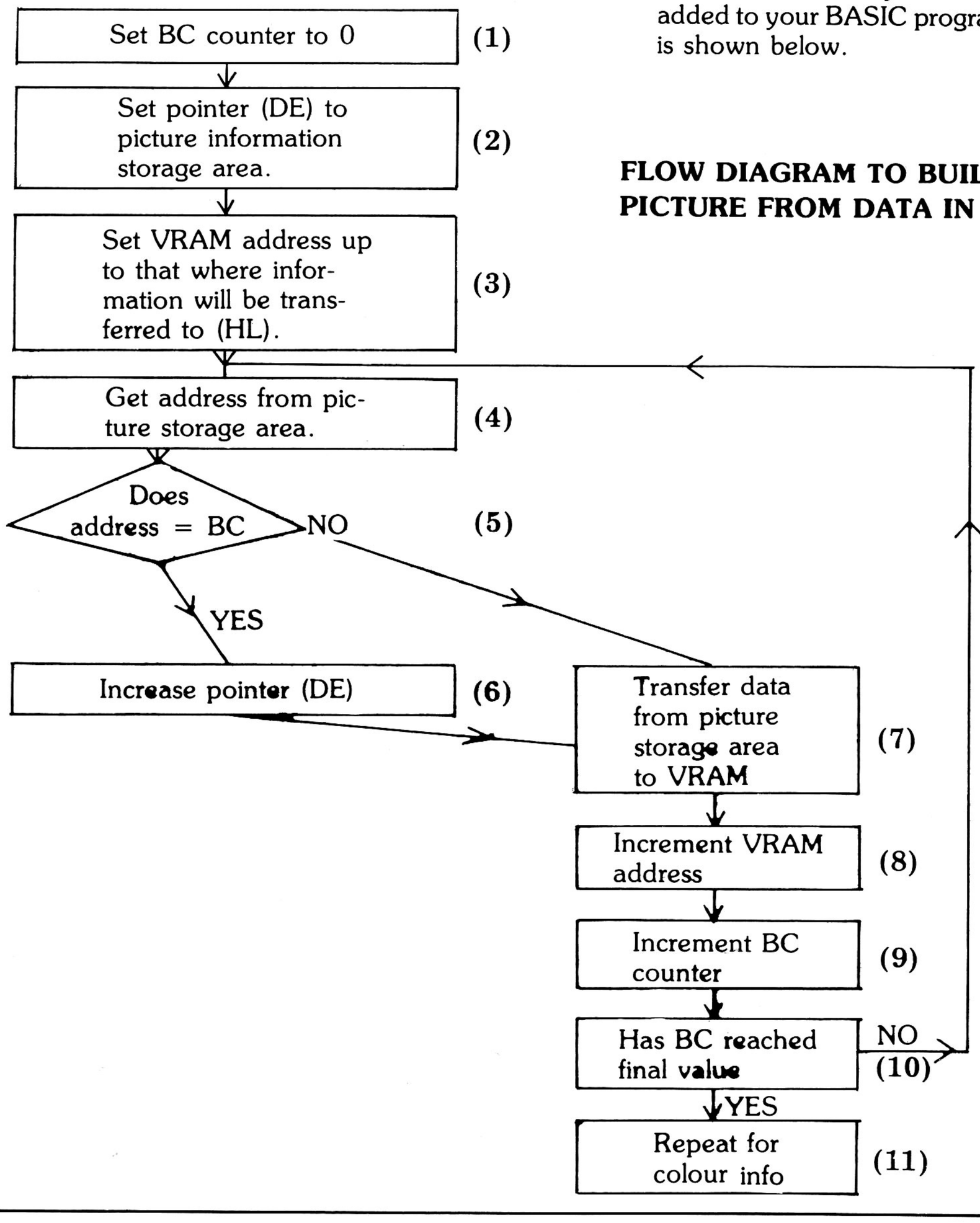
A program to accomplish this routine could be written in BASIC but as there are over 10000 points to sample and analyse the time to this task would be excessive. For this reason the machine code routine was developed.

ROUTINE TO DISPLAY A PICTURE STORED ON TAPE.

The GRAPHIC DESIGNER allows you to store the picture that you have drawn onto tape. The information stored is that from memory location &HB000 onwards as described above.

To enable you to use this picture on your own program, a machine code utility of about 80 bytes in length has to be added to your BASIC program. The associated flow diagram is shown below.

FLOW DIAGRAM TO BUILD UP PICTURE FROM DATA IN RAM.



EXPLANATION OF FLOW DIAGRAM

(1) Counter known as BC is used to keep track of the number of picture elements sampled.

(2) DE pointer is set to memory location where picture information is stored.

(3) VRAM address is set to start of display area (in the case of GRAPHIC DESIGNER to &H0100 as explained previously).

(4) The information from the picture storage area is in groups of three bytes:- the first two bytes give the address and the third gives the pixel information. This step gets the first two bytes from the picture storage area.

(5) Compares the two bytes from (4) with counter (BC). If it is the same then a change in pixel information is required at this address.

(6) If step (5) is the same then picture information is effectivelly increased by three bytes. Next set of information is now available.

(7) Data from picture information area (3rd byte) is transferred to video ram.

(8) VRAM address increased by one. (This is an automatic function of step (7)

(9) BC counter counter is increased

(10) BC counter checked to determine if end of picture is reached.

(11) Procedures (1) to (10) is repeated to produce the colour information for the picture.

MACHINE CODE ROUTINE

The listing below is the machine code routine necessary to accomplish the proceedure above. It can be located anywhere in RAM.

AD00 F5	PUSH AF
AD01 C5	PUSH BC
AD02 D5	PUSH DE
AD03 E5	PUSH HL
AD04 F3	DI
AD05 DBBF	IN A, (BF)
AD07 010000	LD BC,0000
AD0A 1100B0	LD DE,B000
AD0D 210001	LD HL,0100
AD10 CD442C	CALL 2C44
AD13 210000	CALL 2C44
AD16 1A	LD A, (DE)
AD17 B9	CP C
AD18 2009	JR NZ AD23
AD1A 13	INC DE
AD1B 1A	LD A, (DE)
AD1C B8	CP B
AD1D 2027	JR NZ AD46
AD1F 13	INC DE
AD20 1A	LD A, (DE)
AD21 67	LD H, A
AD22 13	INC DE
AD23 7C	LD A, H
AD24 D3BE	OUT (BE), A
AD26 03	INC BC
AD27 3E15	LD A, 15
AD29 B8	CP B
AD2A 20EA	JR NZ AD16
AD2C CB45	BIT 0, L
AD2E 2010	JR NZ AD40
AD30 2C	INC L
AD31 E5	PUSH HL
AD32 210021	LD HL, 2100
AD35 CD442C	CALL 2C44
AD38 E1	POP HL
AD39 2600	LD H,00
AD3B 010000	LD BC,0000
AD3E 18D6	JD AD16
AD40 E1	POP HL
AD41 D1	POP DE
AD42 C1	POP BC
AD43 F1	POP AF
AD44 FB	EI
AD45 C9	RET
AD46 1B	DEC DE
AD47 18DA	AD23

DISPLAYING A PICTURE STORED ON TAPE ONTO YOUR OWN PROGRAM

At last we reach the objective of this article. This is best described in numbered steps followed by an example.

(1) Load your BASIC program, then load the utility program as given in lines 10 to 50 in the example below. This can be added to your BASIC program. Run the utility part of the program so that the machine code is poked into its memory locations. The utility is designed to construct a picture from the data at &HB000 but this can be changed if desired.

(2) Determine the area of free RAM available to store the picture information in. You can determine where the end of your BASIC program is by typing in "PRINT HEX$(PEEK(&H8167)" to give the most significant byte. "PRINT HEX$(PEEK(&H8166))" gives the least significant byte.

(3) It is advisable to set end of memory pointer (&H8168 & &H8169) to a value just below your picture storage area. If this is not done then any alterations to the BASIC program will shift the location of the picture information.

(4) Poke &H8161 & &H8160 with the memory value at which the picture information will be stored. In the example it is &HB000.

(5) Load your saved picture off tape.

(6) Reset pointers &H8161 & &H8160 to their original values. These will usually to be &H98 & &H00 respectively. Now type in "GOTO 40".

(7) If all has gone well then your masterpiece will now appear on the graphics screen.

EXAMPLE

Assumptions

(1) BASIC program located between &H9800 &HA900

(2) Picture information start address &HB000

(3) Machine code routine to start at &HAD00

```
10 RESTORE 20:FOR X=&HAD00 TO &HAD48:READ B$:C$="&H"+B$:C=VAL(C$):POKEX,C:NEXT
20 DATA F5,C5,D5,E5,F3,DB,BF,01,00,00,11,00,B0,21,00,01,CD,44,2C,21,00,00,1A,B9,
20,09,13,1A,B8,20,27,13,1A,67,13,7C,D3,BE,03,3E,15,B8,20,EA,CB,45,20,10,2C,E5,21
,00,21,CD,44,2C,E1,26,00,01,00,00,18,D6,E1,D1,C1,F1,FB,C9,1B,18,DA
30 STOP
40 SCREEN 2,2:CALL&HAD00
50 GOTO 50
```

HOUSEHOLD ALARM PROJECT

With some home burglar alarm systems costing as much as $650.00 this project is a very cheap alternative and costs approximately $100-150. (Depending on how complicated you want it to be and if you to use smoke detectors or pressure mats.) With the right programming this burlar alarm can be as good as any commercial alarm.

This project is designed for those of you who have a general knowledge about electronics. You will be required to build an electronic amplifier, choose a correct power supply for your application, wire a speaker and a series of electronic switches.

It works on the same principal as any other household burglar alarm in that around the building there are switches mounted on the doors and windows etc. When a door is opened it will activate the switch which is detected by the computer and thus activating a loud speaker siren usually palced high up on the outside of the building.

There is one major difference with this alarm system, and it is it is SOFTWARE CONTROLLED. This means that say when the front door is opened you have 30 seconds to get to the computer and enter a password to stop the alarm from sounding and waking up the neighbours.

This project requires an external weather speaker that can be mounted outside so that the alarm can be heard. It would have been easy to make this alarm system work off your T.V. or Hi Fi unit but it wouldn't be practical as you would have to leave your T.V. etc on all day and night and nobody outside could hear the alarm if it was set off. We have to make a small low voltage amplifier so as not to double your power bill!!

With this project all you have to do is keep your SEGA and AMP POWER SUPPLY on.

CONSTRUCTION

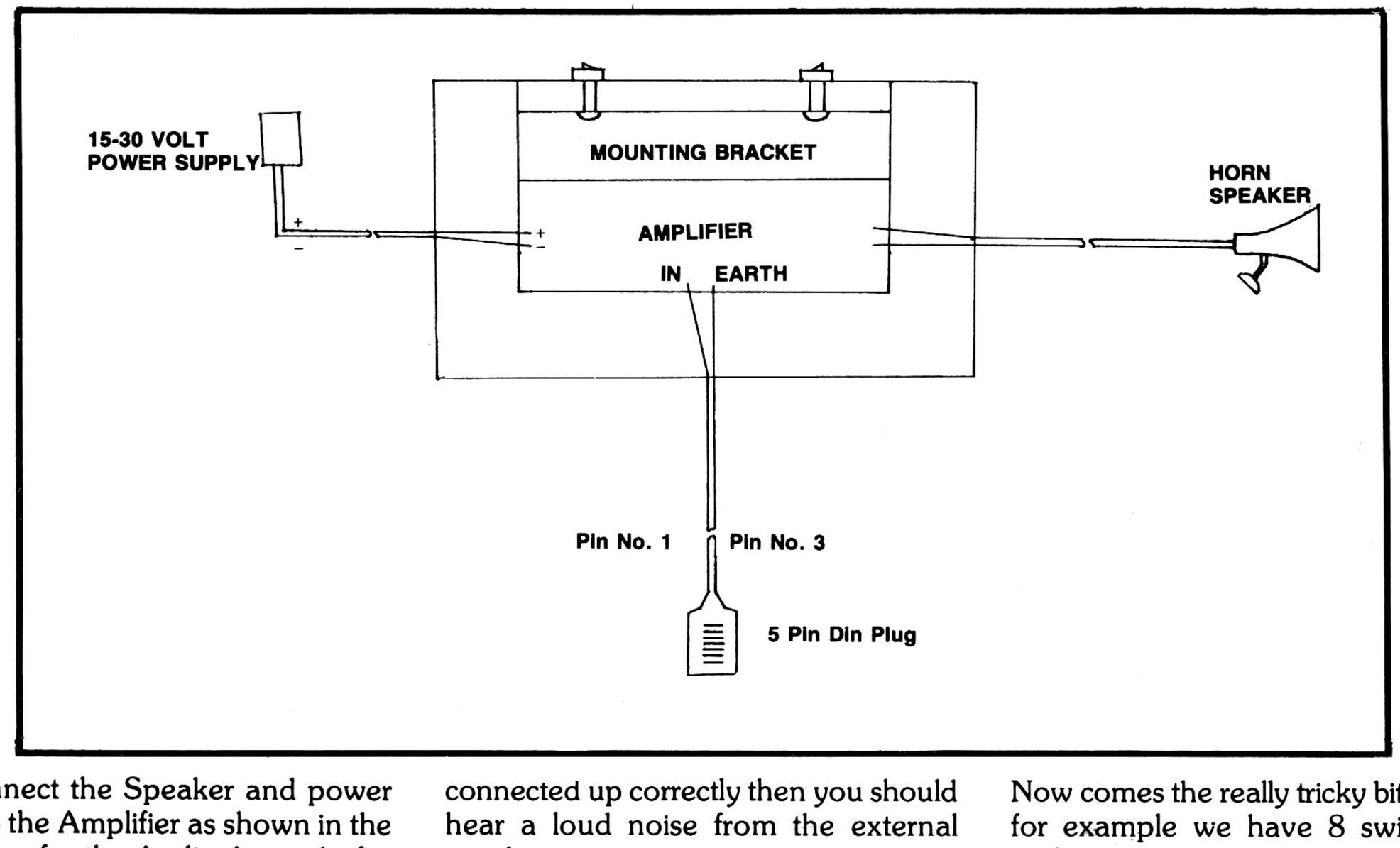
Start with the Audio amplifier, if you already have a general purpose Audio amp then skip this section.

David Reid Electronics have an Audio Amplifier in kit form, this kit is suitable for intermediate to advanced electronic enthusiasts. (See parts list below.)

One small point, if you are going to use this kit we found it necessary to bend the middle "leg" on Q3 transistor over to the middle pin on the P.C.B to get the transistor to fit. (This may have been just our individual kit but watch out for this.)

With the amplifier completed mount the unit against the side of the case aprox 10mm from the bottom.

Drill two small holes adjacent to the mounting bracket and then each side of it. These will be used for the speaker, power supply and the sound input wires as shown on the diagram below.



Now connect the Speaker and power supply to the Amplifier as shown in the instructions for the Audio Amp. As for the "input" side of the amp, connect the "EARTH" wire to pin 3 on Video plug on the back of the computer, then the "IN" wire has to be connected pin 1 on the 5 pin din video socket. Once the amplifier, speaker and power supply have been connected up it is a good idea to plug it all in as directed in fig 1. Then type in "sound 1,110,15" to your SEGA SC-3000, and if every thing was connected up correctly then you should hear a loud noise from the external speaker.

We can now mount the proximity switches on the doors and windows etc. Depending on your program you can have up to 12 of these switches around your house. Try to hide these if possible. Then run a cable from the each of the switches to the computer joystick ports. A MAXIMUM of six switches can be wired to each joystick port.

Now comes the really tricky bit. Lets say for example we have 8 switches we wish to monitor. From each micro switch located on a door or window there is two wires running to the computer. For the first six switches take one wire from each switch and connect it to the "COMMON" pin. (this means that pin No 8 will have 6 wires connected to it. See Diagram 2 below.) As for the other wire, on each micro switch connect them individually to pins 1,2,3,4,6 and 9 on the JOYSTICK 1 port. For the

other two switches hook one wire from each up to the common pin on JOYSTICK 2 port (Pin 8) and the other wire from each micro switch individual to any of the six pins which the computer scans. (Pins 1,2,3,4,6 and 9)

What we have done is to hook each micro switch up to a joystick port pin that the computer scans. A signal is constantly sent out through Pin 8. When a door or window is opened and the circuit is complete and a signal is received from that individual micro switch. By scanning individually pins we can tell exactly which window or door has been opened and correspondingly take action as required. (Eg turn on the alarm and scare the hell out of any unwanted visitors!)

Each pin has direction or fire button assigned to it. (see diagram 2.)

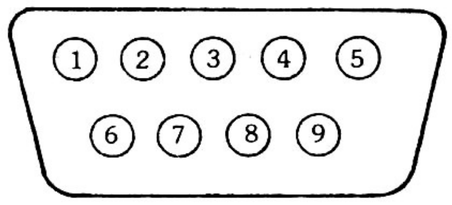
Using both the STICK and STRIG commands for both joystick ports to test for any instructions. For example if you check scanning STICK(1) and we get a value of 1 ("UP") returned we know that the door, window etc attached to Pin 1 has been opened. If a value of Zero is given everything is secure. We have listed a Demonstration programme with eight devices hooked up. (Front and Back door, 4 windows and two smoke detectors.)

1. Joystick Terminal

Two connector sockets, JOY1 and JOY2, for joysticks are supplied at the rear end of the SF-3000, each connected to the P.P.I. inside the SF-3000.

Pin number	Function
1	UP
2	DOWN
3	LEFT
4	RIGHT
5	———
6	Trigger LEFT
7	———
8	Common
9	Trigger RIGHT

AMP 9-pin male type

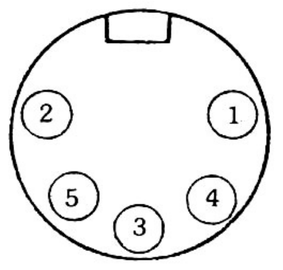


Outlook of the joystick connector socket and its pin numbers

2. Video/audio Output Terminal

Pin number	Function
1	Audio signal
2	Video signal
3	GND
4	GND
5	GND

DIN 5-pin female type



Outlook of the VIDEO connector socket and its pin numbers

See pages 181 and 183 in the SF7000 disk drive manual or pages 149-150 in the level III Sega manual.

PARTS LIST

	Part No.	Price	Available from
1 × UNIVERSAL AUDIO AMPLIFIER	K133	$19.95	David Reid Elec
NOTE: This amplifier is a electronic kit which will require you to perform soldering and assembly. It is reasonably easy to make!			
2 x AMP 9 — pin male type plug	PCTE9P	$4.31	David Reid Elec
NOTE: These are the joystick plugs			
1 x 145mm * 54mm Plastic box.	BP6	$14.70	David Reid Elec
1 x 5 pin din plug	P15	$1.45	David Reid Elec
1 x 8 Ohm SPEAKER HORN 5"	RH5P	$22.95	David Reid Elec
1 x 15-30 volt DC power supply		$30-40	David Reid Elec
NOTE: The greater the voltage of the power supply the louder the alarm. We used a 24v DC and found this quite acceptable.			
2x 3mm x 15mm Nuts and Bolts for mounting the Amp to its case.		$2.99	Dick Reid Elec David Smith Elec
1 x 2.1mm Power socket for adapter	P37	$2.50	David Reid Elec

ASSTD SWITCHES:

WINDOWS: Proximity switches with magnets. (Normally open)	SW80 L-5212	$3.95	David Reid Dick Smith
DOORS: Proximity switches as above			
PRESSURE MAT: Placed under carpet or mat. Activated when Trodden on.	L-5270	$30.00	Dick Smith
HEAT/FIRE SENSORS: (Normally open)	L-5254	$15.00	Dick Smith

PLEASE NOTE: There is a large range of security switches available from most electronic shops, but unless you run them through an inverter then please make sure they are of the "NORMALLY OPEN" type. (Otherwise the alarm will go off all the time and turn off when somebody breaks in!)

TWIN FLEX WIRE: for wiring switches and speaker etc $.50 per metre

As you can see this project opens up, a whole range or possibilities for interpreting data from external devices from the Joystick ports.

Household Burglar Alarm Programme

This is a demonstration programme that scans five doors and windows, two smoke detectors and the front door.

Line 50 checks to see any one of the four normal doors or windows have been opened. This means that all the normal doors and windows have to be hooked up to joystick port 1.

Line 60 checks to see that the smoke detectors have been set off. This means that they must be connected to the fire pins on joystick port 1.

Line 70 checks to see if the front door has been opened. If it has, you have one minute to key in a code (PN$) or else the alarm will be set off. The front door is wired to the up pin on the joystick 2 port. If this all seems confusing just experiment and all should become crystal clear.

```
10 REM HOUSE HOLD ALARM PROJECT
20 REM       BY GRANDSTAND
30 A=STICK(1):B=STRIG(1):C=STICK(2)
40 IF A=0 AND B=0 AND C=0 THEN 30
50 ON A GOTO 30,100,100,100,100,100,100,100,100
60 ON B GOTO 30,200,200,200
70 ON C GOTO 30,300,100,100,100,100,100,100,100
100 REM DOOR OR WINDOW OPENED
110 SOUND 1,440,15:SOUND 2,441,15:SOUND 3,442,15:FOR DE=1 TO 25:NEXT
120 SOUND 1,640,15:SOUND 2,641,15:SOUND 3,642,15:FOR DE=1 TO 25:NEXT
130 GOTO 110
200 REM SMOKE DECTOR ACTIVATED
210 FOR I=880 TO 450 STEP -I/25:SOUND 1,I,15:SOUND 2,I+1,15:SOUND 3,I+2,15:NEXT
220 GOTO 210
300 REM FRONT DOOR OPENED. YOU HAVE
310 REM 1 MIN TO GET TO THE COMPUTER
320 REM AND KEY IN PASS NUMBER TO
330 REM STOP THE ALARM SOUNDING.
340 PN$="1234"
350 TIME$="00:00:00"
355 FOR I=1 TO 4
360 IF TIME$>"00:01:00" THEN 100
370 A$=INKEY$:IF A$="" THEN 360
380 IF A$=MID$(PN$,I,1) THEN 400
390 IF A$<>MID$(PN$,I,1) THEN 100
400 IF INKEY$=A$ THEN 400
410 NEXT
```

Graphics to SP-400

by Michael Howard

Last issue featured a program for copying the picture on the graphics screen to the SP-400 Printer Plotter. Well, I decided to update the program and write it in machine-code so that it would run much quicker. I'd like to take this opportunity to thank G.W. Emms for giving me the original idea. Ta Mate!! All you do for this program is load it and run it! . . . EASY!

```
10 DATA0,0,0,ED,5B,0,F0,CD,4A,51,FE,0,C0,ED,5B,0,F0,1C,AF,BB,20,A,14,3E,C0,BA,20
,4,32,2,F0,C9,ED,53,0,F0,C3,3,F0
20 T=0:FORA=&HF000TO&HF026:READA$:V=VAL("&h"+A$):POKEA,V:T=T+V:NEXT:IFT<>4361THE
NBEEP2:PRINT"ERROR":STOP
30 LPRINTCHR$(18);"A":LPRINTCHR$(18);"M0,-200":POKE&HF000,0:POKE&HF001,0:POKE&HF
002,0:CALL&HF003
40 LPRINT"M";PEEK(&HF000);",";191-PEEK(&HF001):LPRINT"J1,0":CALL&HF00D:GOTO40
```

SEGA COMPUTER SCHOOL

This is GRANSTAND'S latest project. We have converted about 1000sq ft of our Auckland office into a classroom, painted and carpeted it and filled it with 10 SEGA'S. The school will be used to teach young and old, experienced and inexperienced about computers and how to use them. Classes began on June 10th and have been a roaring success with the first months beginners courses being totally booked out. Eventually, we intend to run three separate course (Beginners, Intermediate and Advanced). The beginners course is designed to help new computer owners and people who have little experience on coumputers to learn how to use a computer and what they can do. This course is not brand specific, so if people attend this course, they can transfer what they learn to just about any computer. The Intermediate Course is designed for people who own a SEGA and have had a little experience. It will cover sound and graphics and teach good programme structure and implementation. The Advanced course is for people who want to learn all about things like Machine Code and PEEKing and POKEing. Note: Not recommended for those who don't have at least a PhD in Maths. No seriously, the courses have been well tested and move at a rate determined by the class itself. There are plenty of programmes to type in and to take home to help you fully understand what you have studied that week. Yes, homework is set but not marked as I am to lazy)!

Each course consists of 4 two hour lessons one each week for a month for a charge of $40. The courses are held Monday to Thursday each week between 5.30pm and 7.30pm. If the demand is great, enough courses will be held from 8.00pm to 10.00pm Monday to Thursday and on Saturdays. Weekend and day courses will also be available on selected days and weekends, and during the school holidays.

PLEASE contact GRANSTAND for Further Details and Booking information.

BLACK JACK

This programme makes use of the Sega's excellent graphics to produce an intelligent Black Jack game which can get rather difficult to beat.

This program can be played by up to 4 people

```
1 PATTERNC#79,"708888888888870":PATTERNC#48,"708898A8C8887O":PATTERNC#160,"3C4299
919199423C"
10 REM BLACKJACK BY S.Coupe
11 GOSUB9000
20 F=1:X=104:CR=1
22 IFM(F)>OTHENGOSUB4000:BEEP1:BEEPO
23 IFM(F)=OTHENCURSOR13O,(F-1)*40+20:COLOR4:PRINTCHR$(17);"OUT";CHR$(16)
24 F=F+1:IFF>PLTHEN3O
25 GOTO22
30 GOSUB2000
35 FORF=1TOPL
40 IFM(F)=OTHEN92
45 BLINE(O,170)-(255,191),,BF:CURSOR10,170:PRINTP$(F);" please enter your bet:";
50 GOSUB3000
90 BEEP:B(F)=T:BLINE(52,(F-1)*40+12)-(80,(F-1)*40+19),,BF:CURSOR46,(F-1)*40+12:M
(F)=M(F)-T:PRINTM(F):CURSOR30,(F-1)*40+20:PRINTT
92 NEXTF
95 BLINE(O,170)-(255,191),,BF
100 FORF=1TOPL:IFM(F)=OANDB(F)=OTHEN105
102 X=136:CR=2:GOSUB4000:BEEP1:BEEPO
105 NEXT
110 GOSUB2000
120 F=1:X=168:CR=3
125 IFM(F)=OANDB(F)=OTHEN305
130 COLOR1:CURSOR20,170:PRINTP$(F);" Stay,Twist or Buy ?"
140 I$=INKEY$:IFI$="S"THEN300
150 IFI$="T"THEN200
160 IFI$="B"THEN500
170 GOTO140
200 GOSUB4000:BEEP1:BEEPO:X=X+32:CR=CR+1
210 IFC(F,1)+C(F,2)+C(F,3)+C(F,4)+C(F,5)>21THEN1000
220 IFCR=6THEN8000
230 GOTO140
300 FORI=15TOOSTEP-1:SOUND1,600,I:SOUND1,650,I:SOUND1,700,I:NEXT:BLINE(O,170)-(2
55,191),,BF
305 X=168:CR=3:F=F+1:IFF>PLTHEN320
310 GOTO125
320 A=C(4,1):B=C(5,1):X=104:F=4:GOSUB4005:C(4,1)=A
330 A=C(4,2):B=C(5,2):X=X+32:GOSUB4005:C(4,2)=A
332 TT=C(4,1)+C(4,2)
335 IFC(1,1)=OANDC(2,1)=OANDC(3,1)=OTHEN700
340 CR=3:X=X+32
350 TT=C(4,1)+C(4,2)+C(4,3)+C(4,4)+C(4,5):IFTT>21THEN800
360 IFTT>18THEN700
370 IFTT>16ANDRND(1)>.49THEN700
375 IFCR=6THEN700
380 GOSUB4000:C(4,CR)=A:CR=CR+1:X=X+32:GOTO350
500 BEEP:BLINE(O,170)-(255,191),,BF
510 IFM(F)=OTHENPRINT"You cant afford to buy":FORT=OTO500:NEXT:BLINE(O,170)-(255
,191),,BF:GOTO130
511 CURSOR10,170:PRINT"How much more will you bet:"
520 GOSUB3000
550 B(F)=B(F)+T:M(F)=M(F)-T
555 BLINE(O,170)-(255,191),,BF
560 BLINE(52,(F-1)*40+12)-(80,(F-1)*40+19),,BF:CURSOR46,(F-1)*40+12:PRINTM(F):BL
INE(36,(F-1)*40+20)-(80,(F-1)*40+28),,BF:CURSOR30,(F-1)*40+20:PRINTB(F)
```

```
570 GOTO200
700 IFTT=21THENBLINE(0,170)-(255,191),,BF:CURSOR20,170:COLOR1:PRINT"Banker has b
lackjack.":FORI=1TO4:FORT=200TO330STEP10:SOUND1,T,15:NEXTT,I:GOTO720
710 BLINE(0,170)-(255,191),,BF:CURSOR20,170:COLOR1:PRINT"Banker will stay with";
TT
720 SOUND0:F=1
725 IFC(F,5)>0THEN750
730 IFC(F,1)=0THEN750
732 TL=C(F,1)+C(F,2)+C(F,3)+C(F,4)
734 IFTL=21ANDTT<TLTHENCURSOR20,(F-1)*40+28:PRINT"BLACKJACK":FORI=1TO20:SOUND1,1
000,15:SOUND1,800,15:SOUND0:NEXT:M(F)=M(F)+(B(F)*3):M(4)=M(4)-(B(F)*2):GOTO740
735 IFTL>TTTHENCURSOR60,(F-1)*40+28:PRINT"Won":M(F)=M(F)+(B(F)*2):M(4)=M(4)-B(F)
736 IFTL<=TTTHENCURSOR60,(F-1)*40+28:PRINT"Lost":M(4)=M(4)+B(F)
740 IFM(4)<0THENM(4)=0
741 BLINE(52,132)-(80,139),,BF:CURSOR46,132:PRINTM(4):BLINE(36,(F-1)*40+20)-(80,
(F-1)*40+27),,BF
742 BLINE(52,(F-1)*40+12)-(80,(F-1)*40+19),,BF:CURSOR46,(F-1)*40+12:PRINTM(F)
743 BEEP:IFM(4)=0THEN8500
750 F=F+1:IFF>PLTHEN760
755 GOTO725
760 FORF=0TO300:NEXT:BLINE(0,170)-(255,191),,BF:CURSOR108,170:PRINT"¡ PRESS ANY
KEY ¡"
765 IFINKEY$=""THEN765
770 BEEP:FORF=1TO5:FORT=1TO5:C(F,T)=0:NEXTT,F:FORF=1TO13:FORT=1TO4:PA(F,T)=0:NEX
TT,F
775 IFM(4)<1THEN8500
780 FORF=0TO2:B(F+1)=0:BLINE(0,F*40+28)-(84,F*40+36),,BF:NEXT
785 IFM(4)=8000THEN8700
790 LINE(104,0)-(255,191),11,BF:COLOR1,11,(104,0)-(255,191):BLINE(104,0)-(255,19
1),,BF
795 GOTO20
800 IFC(4,1)=11THENC(4,1)=1:GOTO350
801 IFC(4,2)=11THENC(4,2)=1:GOTO350
802 IFC(4,3)=11THENC(4,3)=1:GOTO350
803 IFC(4,4)=11THENC(4,4)=1:GOTO350
804 IFC(4,5)=11THENC(4,5)=1:GOTO350
809 BLINE(0,170)-(255,191),,BF:CURSOR20,170:COLOR1:PRINT"Banker is bust"
810 FORT=500TO110STEP-10:SOUND1,T,15:NEXT
820 TT=0:GOTO720
1000 IFC(F,1)=11THENC(F,1)=1:GOTO210
1010 IFC(F,2)=11THENC(F,2)=1:GOTO210
1020 IFC(F,3)=11THENC(F,3)=1:GOTO210
1030 IFC(F,4)=11THENC(F,4)=1:GOTO210
1040 IFC(F,5)=11THENC(F,5)=1:GOTO210
1050 RESTORE1060:FORI=1TO12:READA,P:SOUND3,A,0:SOUND5,3,15:FORT=240TO250STEPP+1:
OUT127,T:NEXTT,I
1060 DATA 1760,.5,1568,1,1760,.5,1568,1,1760,.5,1568,1,1397,1,1319,1,1175,1,9999
9,.5,880,.2,880,.1
1070 SOUND0:C(F,1)=0:CURSOR20,(F-1)*40+28:COLOR8:PRINT"BUST":COLOR1
1080 BLINE(36,(F-1)*40+20)-(80,(F-1)*40+27),,BF:BLINE(52,132)-(80,139),,BF:M(4)=
M(4)+B(F):CURSOR46,132:PRINTM(4)
1085 BLINE(0,170)-(255,191),,BF
1090 F=F+1:CR=3:X=168:IFF>PLTHEN320
1100 GOTO125
2000 COLOR1,2,(X,121)-(X+23,153):CURSORX+6,130:PRINTCHR$(17);"?";CHR$(16):BEEP1:
BEEP0
2010 A=INT(RND(1)*13)+1:B=INT(RND(1)*4)+1:IFPA(A,B)=1THEN2010
2020 PA(A,B)=1
2040 C(4,CR)=A:C(5,CR)=B
2050 RETURN
3000 BE$=""
3010 X1=210
3020 IFINKEY$<>""THEN3020
3025 I$=INKEY$
3026 IFI$=CHR$(13)THEN3100
3030 IFI$<"0"ORI$>"9"THEN3025
3040 BE$=BE$+I$:CURSORX1,170:PRINTI$:BEEP:X1=X1+7:IFX1=238THEN3100
3060 GOTO3020
3100 IFVAL(BE$)=0THENBLINE(170,170)-(200,178),,BF:GOTO3000
3110 T=VAL(BE$):IFT<=M(F)THENRETURN
```

```
3120 CURSOR210,180:PRINT"To much":BEEP2:FORI=1TO500:NEXT:BLINE(210,170)-(255,191
),,BF:GOTO3000
4000 A=INT(RND(1)*13)+1:B=INT(RND(1)*4)+1:IFPA(A,B)=1THEN4000
4005 C=1:IFB=1ORB=2THENC=8
4006 D$=CHR$(246):IFB=2THEND$=CHR$(247)
4007 IFB=3THEND$=CHR$(245)
4008 IFB=4THEND$=CHR$(248)
4010 PA(A,B)=1:COLORC,15,(X,(F-1)*40+1)-(X+23,(F-1)*40+33)
4015 A$(1)="":A$(2)="":A$(3)="":A$(4)=""
4016 IFF=4THENBLINE(X,(F-1)*40+1)-(X+23,(F-1)*40+33),,BF
4020 ONAGOTO4200,4210,4220,4230,4240,4250,4260,4270,4280,4290,4300,4310,4320
4030 CURSORX+2,(F-1)*40+1:PRINTA$(1):CURSORX+2,(F-1)*40+9:PRINTA$(2):CURSORX+2,(
F-1)*40+17:PRINTA$(3):CURSORX+2,(F-1)*40+25:PRINTA$(4)
4035 IFF=4THENRETURN
4040 C(F,CR)=A:RETURN
4200 CURSORX+6,(F-1)*40+8:PRINTCHR$(17);"A":CURSORX+6,(F-1)*40+20:PRINTD$;CHR$(1
6)
4201 IFF=4THENA=11:RETURN
4202 C(F,CR)=11:RETURN
4210 A$(2)=" "+D$:A$(4)=A$(2):GOTO4030
4220 A$(2)=" "+D$:A$(3)=A$(2):A$(4)=A$(3):GOTO4030
4230 A$(2)=D$+" "+D$:A$(4)=A$(2):GOTO4030
4240 A$(2)=D$+" "+D$:A$(3)=" "+D$:A$(4)=A$(2):GOTO4030
4250 A$(2)=D$+" "+D$:A$(3)=A$(2):A$(4)=A$(2):GOTO4030
4260 A$(2)=D$+" "+D$:A$(3)=D$+D$+D$:A$(4)=A$(2):GOTO4030
4270 B$=D$+" "+D$:A$(1)=B$:A$(2)=B$:A$(3)=B$:A$(4)=B$:GOTO4030
4280 A$(1)=D$+" "+D$:A$(2)=A$(1):A$(3)=D$+D$+D$:A$(4)=A$(1):GOTO4030
4290 A$(1)=D$+" "+D$:A$(2)=D$+D$+D$:A$(3)=A$(2):A$(4)=A$(1):GOTO4030
4300 A$(1)=" J ":A$(3)=" "+D$:A=10:GOTO4030
4310 A$(1)=" Q":A$(3)=" "+D$:A=10:GOTO4030
4320 A$(1)=" K":A$(3)=" "+D$:A=10:GOTO4030
8000 RESTORE8010:FORI=1TO23:READF2,F3,F1,D:SOUND2,F2,9:SOUND3,F3,9:SOUND1,F1,9:F
ORDE=1TOD/5:NEXTDE,I
8010 DATA 932,1864,165,30,784,1568,311,15,932,1864,311,15,932,1864,330,15,784,15
68,330,15,932,1864,330,15,784,1568,330,15
8020 DATA 698,1397,349,15,932,1864,349,15,1175,2350,349,15,1397,2794,349,15,1397
,2794,175,15,1175,2350,175,15
8030 DATA 932,1864,175,15,698,1397,175,15,784,1586,262,240,932,1864,262,240,1175
,2350,175,120,1047,2094,175,240,932,1864,175,120,932,1864,233,320,932,1864,175,3
20,932,1864,117,720
8040 SOUNDO:M(F)=M(F)+(B(F)*4):M(4)=M(4)-(B(F)*3)
8045 IFM(4)<0THENM(4)=0
8046 BLINE(0,170)-(255,191),,BF
8050 COLOR1:BLINE(52,132)-(80,139),,BF:CURSOR46,132:PRINTM(4):BLINE(36,(F-1)*40+
20)-(80,(F-1)*40+27),,BF
8060 BLINE(52,(F-1)*40+12)-(80,(F-1)*40+19),,BF:CURSOR46,(F-1)*40+12:PRINTM(F)
8065 IFM(4)=0THEN8500
8070 CR=3:F=F+1:X=168:IFF>PLTHEN300
8080 GOTO125
8500 FORI=1TO20:FORF=500TO1000STEP100:SOUND1,F,15:NEXTF,I
8505 SOUNDO
8510 SCREEN1,1:CLS:PRINT" Well done.You managed to make the      banker go bankru
pt.You win.":GOTO8720
8700 SCREEN 1,1:CLS:PRINT"*****************************************  G
AME OVER *****************************************"
8710 PRINT:PRINT"Tough luck,the banker won."
8720 PRINT:PRINT"Try again (y/n) ?"
8730 IFINKEY$="Y"THENCALL&H6C37
8740 IFINKEY$="N"THENEND
8750 GOTO8730
9000 SCREEN2,2:COLOR1,15,,1:CLS
9005 CURSOR20,10:PRINT"Welcome to the game of...":LINE(15,18)-(180,18),8:LINE(17
,20)-(178,20),8
9010 COLOR1:CURSOR70,40:PRINTCHR$(17);"BLACKJACK";CHR$(16)
9020 COLOR8,11,(64,36)-(183,50):LINE(64,36)-(183,50),3,B
9030 COLOR8:CURSOR20,180:PRINTCHR$(160);" GRANDSTAND                    BY S.COUPE"
9040 COLOR1:CURSOR50,90:PRINT"Do you want instructions ?"
9050 IFINKEY$="Y"THEN9200
9060 IFINKEY$="N"THEN9100
9070 GOTO9050
```

```
9100 BEEP:CURSOR40,120:PRINT"How many players ?(1-3)"
9110 I$=INKEY$:IFI$<"1"ORI$>"3"THEN9110
9120 PL=VAL(I$)
9130 SCREEN1,1:CLS:COLOR1,11
9140 FORF=1TOPL:PRINT"Player";F;"s name ? (max 10 letters)"
9150 INPUTN$:IFLEN(N$)>10THENN$="SOME TWIT"
9155 P$(F)=N$
9160 NEXT
9170 COLOR1,3:CLS:GOTO9500
9200 BEEP:BLINE(50,90)-(210,100),,BF
9210 CURSOR10,60:PRINT"  This is a card game where you have to  get as close to
21 as you can.If you get 21 then that is Blackjack.If you get over       21 then
 you are bust."
9220 PRINT:PRINT"   All players are playing against the    banker,played by the
SEGA.You must try  and make the banker go bankrupt.To win a  round you must have
 a higher score than the banker.If you have the same you lose."
9230 CURSOR90,160:PRINT"PRESS ANY KEY":IFINKEY$=""THEN9230
9240 BEEP:BLINE(0,60)-(255,170),,BF
9250 CURSOR10,60:PRINT"If you get an ace it can be counted as 1 or 11.Kings,Quee
ns and jacks are all 10s.":PRINT:PRINT"  If you get 5 cards that add to 21 or
     under you win 3 times your bet."
9260 PRINT:PRINT"  A blackjack will win you 2 times your  bet,as long as the ban
ker hasn't got 21!"
9270 CURSOR90,160:PRINT"PRESS ANY KEY":IFINKEY$=""THEN9270
9280 BEEP:BLINE(0,60)-(255,170),,BF
9290 CURSOR10,60:PRINT" When given the chance you can stay,buy   or twist.Stay m
eans you don't want any    more cards.Buy means you want to bet     some more mo
ney and get another card."
9300 PRINT"  Twist means you get another card but       don't want to bet any mo
re money.":GOTO9100
9400 STOP
9500 DIMPA(13,4),C(5,5)
9505 M(1)=1000:M(2)=1000:M(3)=1000:M(4)=5000
9510 SCREEN2,2:COLOR1,11,,5:CLS:COLOR1,15,(0,0)-(95,162):LINE(0,0)-(95,162),1,B
9520 FORF=0TO160STEP40:LINE(0,F)-(95,F+2),,B:NEXT
9530 X=4:FORF=1TOPL:CURSOR10,X:PRINTP$(F):CURSOR10,X+8:PRINT"Money:$1000":CURSOR
10,X+16:PRINT"Bet:":X=X+40:NEXT
9540 CURSOR10,124:PRINT"The Banker.":CURSOR10,132:PRINT"Money:$5000"
9550 RETURN
```

ERROR MESSAGES

Cont. from p. 25

than that used with the DIM statement in the beginning of the program. (What a mouth full) To correct this increase the value of the number inside the brackets for that array where it is DIMed. If the subscript (the thing inside the brackets) is a variable you will have to check back through the program looking for the lines that use that variable. Check these lines as if there is an error here it will carry on through and muck the program up here and later.

REDIM'D ARRAY ERROR: You have DIMensioned an array twice (Heinous crime! Smack Smack . . .) This will probably happen when you are running a program for a second time. To correct this place an ERASE before the first of the DIM statements in the program. Also list the offending line in case you have made a typing error. (If your anything like me this is more than likely!)

SYSTEM ERROR: You have really done it now! This EM's is very uncommon. The only way to correct them is to turn the machine off as you will find that the computer will not function properly. If you are using a Disk Drive this EM can be caused by a power surge. Try running your disk drive of a single wall plug instead or having all or your computer gear on double adaptors or a single junction box.

The only way to prevent errors from occuring is to be careful. If you are creating your own programs make a plan of what you intend to do and which variables and lines you are going to use before you start. Also when performing modifications be sure to erase all the old lines. In last months magazine we showed you how to redefine the Zero. Type in the following to redefine the capital i:

PATTERN C#73, "F82Ø2Ø2Ø2Ø2Ø F8ØØ"

More on EM's next issue.

SOFTWARE!

AVAILABLE

CARTRIDGES	PRICES
□BASIC LEVEL IIIB	$175
□MUSIC	$125
□N/SUB	$39.95
□SAFARI HUNTING	$39.95
□BORDER LINE	$39.95
□CONGO BONGO	$39.95
□YAMATO	$39.95
□STAR JACKER	$39.95
□CHAMPION TENNIS	$39.95
□MONACO GP	$39.95
□CHAMPION BASEBALL	$39.95
□SINBAD MYSTERY	$39.95
□VIDEO FLIPPER	$39.95
□PACAR	$39.95
□POP FLAMER	$39.95
□CHAMPION GOLF	$49.95
□EXERION	$49.95
□SAFARI RACE	$49.95
□LODE RUNNER	$49.95
□BOXING	$54.95
□ORGUS	$49.95
□FLICKY	$59.95

HOME/PERSONAL/FINANCE CASSETTES	PRICE
□FILE SYSTEM	$19.95
□CHEQUE BOOK RECONCILIATION	$19.95
□LOAN & MORTGAGE CALCULATION	$19.95
□ACCOUNTS RECEIVABLE	$30.00
□ACCOUNTS PAYABLE	$30.00
□32K WORD PROCESSOR	$30.00
□PERSONAL RECORD KEEPER	$30.00
□16K EASY WRITER	$19.95

SEGA®

- **32K COMPUTER $495**
- **DISK DRIVE $995**
- **JOYSTICK (NEW) $45.00**
- **DATASETTE $89.95**
- **PRINTER $495**

GAME CASSETTES	PRICE
□TOWERS OF HANOI	$19.95
□HANGMAN	$19.95
□CITY-LANDER	$19.95
□CUBE-IT	$30.00
□BUGALOO	$19.95
□LASER BLAST	$19.95
□MUNCHMAN	$30.00
□ENTERPRISE ESCAPE	$19.95
□MARSMOBILE	$19.95
□MARS ADVENTURER	$19.95
□EMPIRE	$19.95
□KALAH	$19.95
□CHESS	$19.95
□DEATH SATELLITE	$19.95
□BIT BYTE	$19.95
□THE HOUSE	$30.00
□THE PYRAMID	$19.95
□CODE BREAKER	$19.95
□BACKGAMMON	$19.95
□LUMBER JACK	$19.95
□TRAMPMAN	$19.95
□SEGA GAMES PACK I	$19.95

EDUCATIONAL CASSETTES	PRICE
□ROCKET MATHS	$19.95
□LEARNING ALPHABET	$19.95
□WATCH ME DRAW	$19.95
□SPELLING	$19.95
□ADDITION	$19.95
□SUBTRACTION	$19.95
□MULTIPLICATION	$19.95
□SPRITE EDITOR	$19.95
□MUSIC EDITOR	$19.95
□TYPING TUTOR	$19.95
□LEARNING TO COUNT	$19.95
□SHAPE & COLOUR QUIZ	$19.95
□MUSIC CARTRIDGE DEMONSTRATOR	$19.95
□SEGA GRAPHIC DESIGNER	$19.95
□GEOGRAPHY QUIZ I	$19.95
□GEOGRAPHY QUIZ II	$19.95
□MATHS HANG UP	$19.95

WRITE FOR OUR COMPLETE PRICE LIST

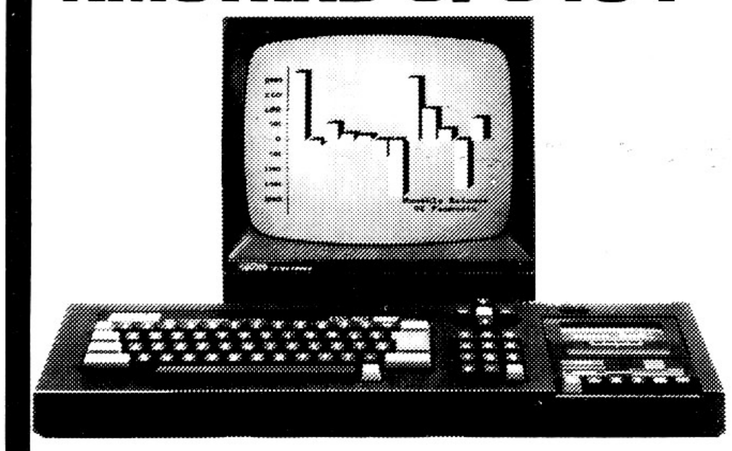
manukau COMPUTERS (N.Z.) LTD.

PHONE AK 656-002

P.O. Box 26-074 AUCKLAND 3

CORNER MANUKAU & PAH RDS - EPSOM

AMSTRAD CPC464



- **GREEN SCREEN $995**
- **COLOUR SCREEN $1395**
- **DISK DRIVE $795**
- **2nd DISK DRIVE $595**

YOUR SEGA COMPUTER IS WELCOME AS A TRADE-IN ON THE NEW AMSTRAD COMPUTER!

I ENCLOSE PAYMENT:

□CASH □CHEQUE □POSTAL ORDER □VISA □BANKCARD

NAME _____

ADDRESS _____

CREDIT CARD NUMBER _____ EXP. DATE _____

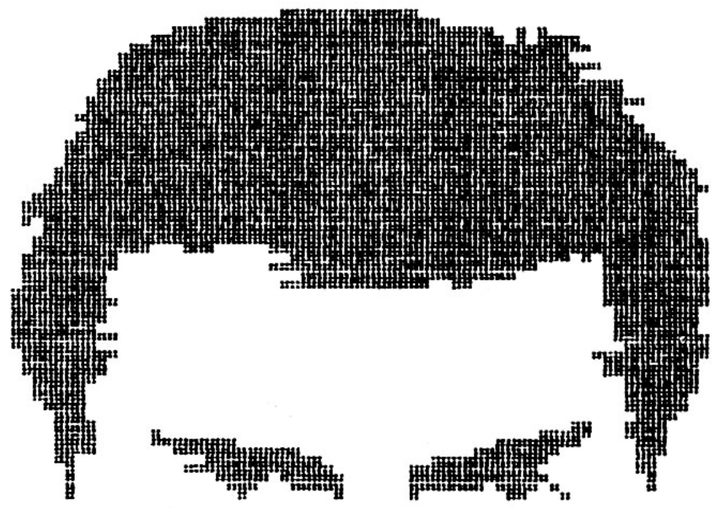
SOFTWARE TITLE(S) REQUESTED _____ PRICE: _____

□AMSTRAD GREEN SCREEN @$995
□AMSTRAD COLOUR SCREEN@$1395
□AMSTRAD DISK DRIVE@$795
□AMSTRAD (SECOND) DISK DRIVE @$595
□SEGA 32K COMPUTER $495
□SEGA DISK DRIVE $995
□JOYSTICK (NEW) $45.00
□SEGA PRINTER $495
□SEGA DATASETTE $89.95

IF YOU DO NOT WISH TO DESTROY THIS MAGAZINE BY TEARING OUT THE COUPON, SEND ANY FACSIMILE OF THIS COUPON WITH ALL THE NECESSARY INFORMATION TO MANUKAU COMPUTERS. BE SURE TO INCLUDE YOUR CREDIT CARD NUMBER AND EXPIRY DATE FOR VISA & BANKCARD PURCHASES.

Freddie

This is another of those graphic demonstrations that we like to fill the magazine with. For all you Queen fans here is Freddie Mercury. The first program draws the picture using sprite, whereas the second program uses redefined characters to get the same result. You will notice the first program using sprites is actually faster drawing the face on the screen.



```
10 REM * FREDDIE *
20 RESTORE 40
30 FOR I=0 TO 76:READ X$:PATTERN S#I,X$:NEXT
40 DATA 00,010103030703070F,00000030F3F7F7FFF,FFFFFFFFFFFFFFFF,007FFFFFFFFFFFFF,F
FFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF,F8FEFFFFFFFFFFFF,FFFFFFFFFFFFF
FFF,0000F2FAFFFFFFFF,FFFFFFFFFFFFFFFF,00000F0F8E0FCF0,FFFFFFFFFFFFFFFF,00,0000C
00080C0E0F0
50 DATA 0F0F1F1F3F7FFFFF,BF3F7F7FFFFFFFFF,FFFFFFFFFFFFFFFF,FFFFFFF18080,FFFFFFFF
FFFFFFFF,FFFFFFC10001,FFFFFFFFFFFFFFFF,FFFFFFF7FFF3FFF,FFFFFFFFFFFFFFFF,FFFFFFF
FFFFFFFFC,FFFFFFFFFFFFFFFF,FFFFFFFFFFFCC0,FFFFFFFFFFFFFFFF,FFFFFFFEB0303003,F8FC
FEFEFFFEFEFE
60 DATA FEFEFFFFFFFFFFFF,FFFFFFFFFEFFFEFF,FFFF7E7C7C3C1C1E,00800000000800080,00,0
3030101010100007,0303,FFFFFEFEFEFFFFFF,FEFEFEFEFCFC7880,1E1E1E1808080808,00,080F0
30001,00,00F0FF7FFF3E1C08,00,0000C0F0F8FC7C04,00,00000000F0F0F0F08,00,000F3FFFFFF
FEF06,00
70 DATA 18F8F0C080,00,F0F07070606020020,00,00,0000000071F1F3F,00,00000002FFFFFFF
F,00,00000010FFFFFFFF,00,000000080C0E0F0,3F3F7F7810,0000000080402010,FFFFFFFF,0
0,FFFFFFFF01,00,F0F8FCFE9C,00,080C0603,00,00003FFF,00,00F0FEF8
80 SCREEN2,2:CLS:MAG1
90 FOR J=0 TO 19:READX,Y:SPRITE J,(X,Y),J*4,1:NEXT
100 DATA 96,42,112,42,128,42,144,42,96,58,112,58,128,58,144,58,96,74,144,74,96,9
0,112,90,128,90,144,89,112,106,128,106,112,122,128,122,112,138,128,138
110 LINE (143,95)-(146,97),1
120 LINE (95,74)-(95,80),1
130 GOTO 130
```

```
10 REM * FREDDIE *
20 RESTORE 40
30 FOR I=128 TO 207:READ X$:PATTERN C#I,X$:NEXT
40 DATA 00,010103030703070F,00000030F3F7F7FFF,FFFFFFFFFFFFFFFF,007FFFFFFFFFFFFF,F
FFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF,FFFFFFFFFFFFFFFF,F8FEFFFFFFFFFFFF,FFFFFFFFFFFFF
FFF,0000F2FAFFFFFFFF,FFFFFFFFFFFFFFFF,00000F0F8E0FCF0,FFFFFFFFFFFFFFFF,00,0000C
00080C0E0F0
50 DATA 0F0F1F1F3F7FFFFF,BF3F7F7FFFFFFFFF,FFFFFFFFFFFFFFFF,FFFFFFF18080,FFFFFFFF
FFFFFFFF,FFFFFFC10001,FFFFFFFFFFFFFFFF,FFFFFFF7FFF3FFF,FFFFFFFFFFFFFFFF,FFFFFFF
FFFFFFFFC,FFFFFFFFFFFFFFFF,FFFFFFFFFFFCC0,FFFFFFFFFFFFFFFF,FFFFFFFEB0303003,F8
FCFEFEFFFEFEFE
60 DATA FEFEFFFFFFFFFFFF,FFFFFFFFFEFFFEFF,FFFF7E7C7C3C1C1E,00800000000800080,00,0
3030101010100007,0303,FFFFFEFEFEFFFFFF,FEFEFEFEFCFC7880,1E1E1E1808080808,00,080F0
30001,00,00F0FF7FFF3E1C08,00,0000C0F0F8FC7C04,00,00000000F0F0F0F08,00,000F3FFFFFF
FEF06,00
70 DATA 18F8F0C080,00,F0F07070606020020,00,00,0000000071F1F3F,00,00000002FFFFFFF
F,00,00000010FFFFFFFF,00,000000080C0E0F0,3F3F7F7810,0000000080402010,FFFFFFFF,0
0,FFFFFFFF01,00,F0F8FCFE9C,00,080C0603,00,00003FFF,00,00F0FEF8,00,00,00
80 SCREEN 2,2:CLS:COLOR 1
90 FOR J=128 TO 204 STEP 4:READ X,Y:CURSOR X,Y:PRINT CHR$(J):CURSOR X,Y+8:PRINT
CHR$(J+1):CURSOR X+8,Y:PRINT CHR$(J+2):CURSOR X+8,Y+8:PRINT CHR$(J+3):NEXT
100 DATA 96,42,112,42,128,42,144,42,96,58,112,58,128,58,144,58,96,74,144,74,96,9
0,112,90,128,90,144,89,112,106,128,106,112,122,128,122,112,138,128,138
110 LINE (143,95)-(146,97),1
120 LINE (95,74)-(95,80),1
130 GOTO 130
```

Delta Race

```
10  MAG1:GOTO810
20  REM MISSILE
30  PATTERNS#100,"0101010103030307"
40  PATTERNS#101,"07040C14070408080D"
50  PATTERNS#102,"80808080C0C0C0E0"
60  PATTERNS#103,"60203028E0201050"
70  PATTERNS#40,"0A0808080D4A6D7F39"
80  PATTERNS#41,"1C0A0D0403030505"
90  PATTERNS#42,"B0202050B256FE9C"
100  PATTERNS#43,"3850B020C0C0A0A0"
110  REM EXPLOSION
120  PATTERNS#252,"830CB42248608BE0"
130  PATTERNS#253,"9080685AC314C03"
140  PATTERNS#254,"C4B06D458A52050 3"
150  PATTERNS#255,"0105022A844C30CA"
160  SCREEN2,2:CLS
170  Z=95
180  COLOR7,5,(0,0)-(255,191),7
190  PATTERNS#0,"0103050913331F4F"
200  PATTERNS#1,"43E7EDFDFB4FE40E"
210  PATTERNS#2,"80C0A090C8CCF8F2"
220  PATTERNS#3,"C2E7B7BFDFF22770"
230  FORT=0TO255
240  IFRND(1)>.5THENSL=INT(RND(1)*3)-1
250  IFZ>140THEN240
260  Z=Z+SL
270  PSET(T,Z),7
280  NEXT
290  LINE(0,150)-(255,150),1
300  PAINT(0,120),7
310  BLINE(0,150)-(255,191),,BF
320  M=-20:S=1
330  FORV=150TO156STEPS
340  S=S+V-150
350  LINE(0,S+V)-(127,150),1:NEXT
360  X=INT(RND(1)*255)
370  LINE(0,150)-(255,150),1
380  FORR=20TO140STEP10
390  LINE(127,150)-(R+M,191),1
400  M=M+10:NEXT
410  M=-20:S=1
420  FORV=150TO156STEPS
430  S=S+V-150
440  LINE(255,S+V)-(127,150),1:NEXT
450  Y=3:ST=172
460  FORQ=186TO191
470  ST=ST+1
480  IFQ<190THENTR=189
490  IFQ>190THENTR=ST
500  LINE(BT,Q)-(TR,Q),1
510  LINE(191,Q)-(ST+20,Q)
520  NEXT
```

```
540 LINE(176,191)-(194,191),1
550 SPRITE 31,(150,160),100,15:SPRITE 30
,(150,176),40,15
555 Y=4:TIME$="00:00:00"
560 PL=STICK(1)
570 IFPL=3THENN=1:A=0:SOUND4,0,15
580 IFPL=7THENN=-1:A=0:SOUND4,0,15
590 IFPL=5THENB=B+1
600 IFPL=1THENB=B-1
610 IFB<0THENSOUND4,2,10
620 IFB>0THENSOUND0
630 IFTIME$>"00:00:30"THEN690
640 A=A+N/10:X=X+A:B=B+.1:Y=Y+B
650 SPRITE0,(X,Y),0,15
660 IFY<2THEN690
670 IFY>180THEN685
675 IFX<40RX>245THEN690
680 GOTO560
685 IFX<1750RX>195THEN690
686 GOTO560
690 SPRITE 0,(X,Y),252,8
700 OUT127,228
710 FORT=240TO255:OUT127,T
720 FORR=1TO15:NEXTR,T
730 PRINT "      DEAD.....":FOR DE=1 TO 300
:NEXT DE
740 GOTO 810
750 SCREEN 1,1:CLS
760 PRINT"CONGRATULATIONS!!
770 PRINT"-You have saved the world"
780 PRINT:PRINT"PRESS FIRE BUTTON TO PLA
Y"
790 IFSTRIG(1)=0THEN790
800 GOTO10
810 SCREEN1,1:CLS
820 PRINT" URGENT MESSAGE.....DATE 19/
12/3060"
830 PRINT"AN EXCAVATION IN THE GREAT NOR
THERN"
840 PRINT"DESERT HAS TRIGGERED AN ANCIEN
T 20TH"
850 PRINT"CENTURY NUCLEAR MISSILE. IT IS
 DUE"
860 PRINT"TO LAUNCH IN 30 SECONDS."
870 PRINT"IN ORDER TO DETONATE THE MISSI
LE AND"
880 PRINT"SAVE THE EARTH, YOU MUST ENTER
 THE"
890 PRINT
900 PRINT"...GOOD LUCK!"
910 PRINT:PRINT"PRESS FIRE BUTTON TO PLA
Y"
920 IFSTRIG(1)<>0THEN20
930 GOTO920
940 SCREEN1,1:CLS:PRINT"THE 30 SECONDS I
S UP!"
950 PRINT"PITY ABOUT THE EARTH"
960 PRINT:PRINT"PRESS FIRE BUTTON TO PLA
Y"
970 IFSTRIG(1)=0THEN970
980 GOTO10
```

FLEXI-GRAPH

This program lets you zoom in on any part of a graph you wish to see — this program is invaluable to anyone like myself who studies mathematics.

Before operating this program make sure you write in an equation in line 310. You can write in any equation from the simple to the complex:

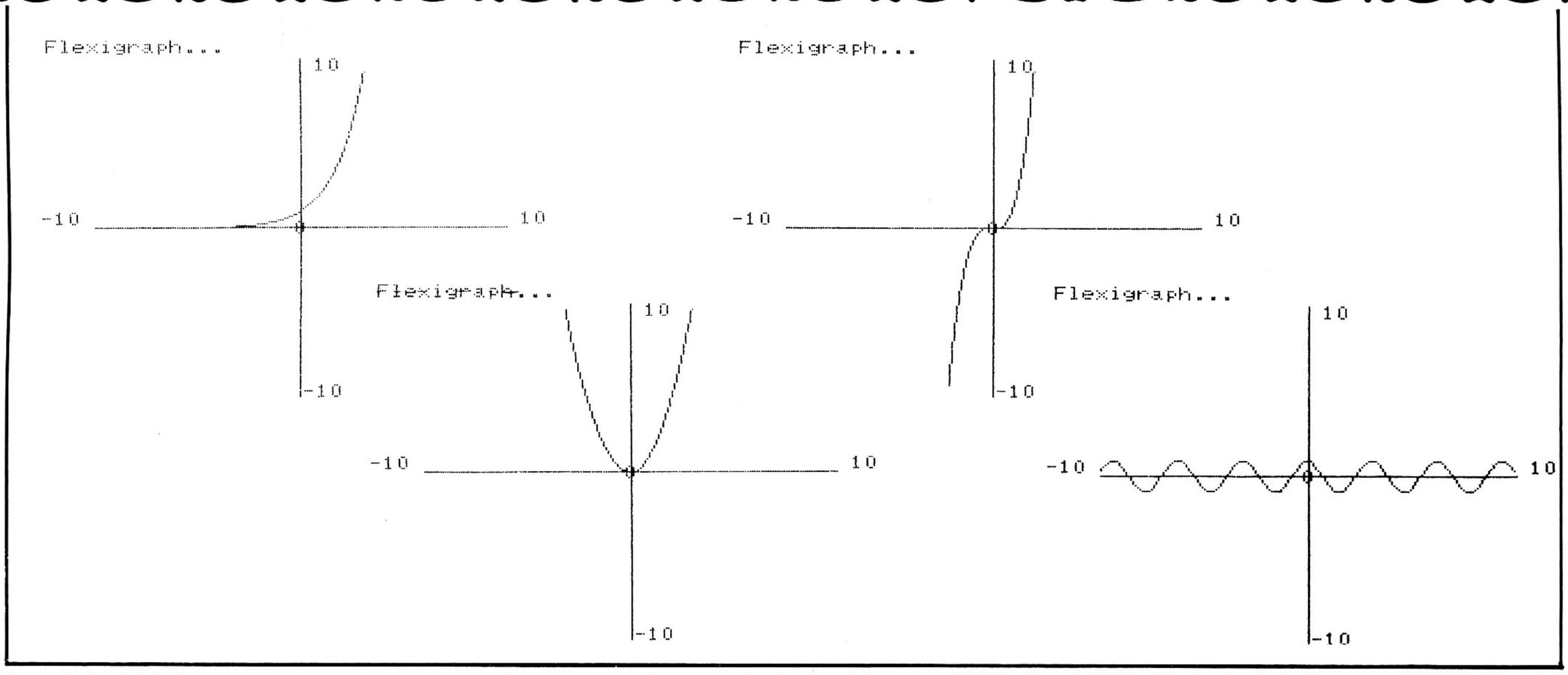
Try writing these:

$y = 2^x$ — Y = 2 X
$y = \cos(x)$ — Y = cos (X)
$y = 1/x$ — Y = 1/X

When you are required to input x min and x max this means from which values of x to which, do you wish to see, similarily y.

One rule: You must have the origin (o,o) where the axes cross in the field of view so as you are able to see where you are relative to the scale on the axes. Therefore you cannot have two positive x values for max and min but you can have one zero or one negative and one positive, the y values must also be done in the same way.

```
10 INPUT "DOMAIN X MIN...";A
20 INPUT "DOMAIN X MAX...";B
30 INPUT "DOMAIN Y MAX...";C
40 D=C-(B-A):J=B-A:I=C-D
50 SCREEN 2,2:CLS
60 PRINT "  Flexigraph..."
70 COLOR 3,3,(0,0)-(255,191),3
80 IF A>B GOTO 10
90 IF A<0 AND B<0 GOTO 10
100 IF A+B=0 THEN 130
110 M=A/(A+B):N=B/(A+B)
120 GOTO 140
130 M=1:N=1
140 G=(220*ABS(M)+40*ABS(N))/(ABS(M)+ABS(N))
150 LINE (G,10)-(G,190),1
160 CURSOR G,10:PRINT C
170 CURSOR G+1,184:PRINT D
180 IF C>0 AND D>0 GOTO 10
190 IF C<0 AND D<0 GOTO 10
200 IF C+D=0 GOTO 230
210 R=C/(C+D):S=D/(C+D)
220 GOTO 240
230 R=1:S=1
240 H=(190*ABS(R)+10*ABS(S))/(ABS(R)+ABS(S))
250 LINE (40,H)-(220,H),1
260 CURSOR G-3,H-3:PRINT"0"
270 IF A<>0 THEN CURSOR 40-(INT(LGT(ABS(A))+1)*12),H-8:PRINT A
280 CURSOR 220,H-8:PRINT B
290 POSITION (G,H),0,1:T=0
300 FOR X=A TO B STEP J/180
310 IF X=0 GOTO 360
320 REM ***************************
        * Put your FORMULA here *
        ***************************
330 IF Y>C OR Y<D GOTO 360
340 IF T=0 THEN PSET (X*180/J,Y*180/I),1:T=1:GOTO 360
350 LINE -(X*180/J,Y*180/I),1
360 NEXT X
370 GOTO 370
```



GLOSSARY

Accessory Devices - additional equipment which attaches to the computer and extends its functions and capabilities. Included are preprogrammed cartridges* and units which send, receive or store computer data, such as printers and disks. These are often called peripherals.

Array - A collection of numeric or string variables, arranged in a list or matrix for processing by the computer. Each element in an array is referenced by a subscript* describing its position in the list.

ASCII - The American Standard Code for Information Interchange, the code structure used internally in most personal computers to represent letters, numbers, and special characters.

BASIC - an easy-to-use popular programming language used in most personal computers. The word BASIC is an acronym for "Beginners All purpose Symbolic Instruction Code."

Baud - commonly used to refer to bits per second.

Binary - a number system based on two digits, 0 and 1. The internal language and operations of the computer are based on the binary system.

Branch - a departure from the sequential performance of program statements. An unconditional branch causes the computer to jump to a specified program line every time the branching statement is encountered. A conditional branch transfers program control based on the result of some arithmetic or logical operation.

Breakpoint - a point in the program specified by the STOP command where program execution can be suspended. During a breakpoint, you can perform operations to help you to locate program errors. Program execution can be resumed with a CONT command, unless editing took place while the program was stopped.

Bug - a hardware defect or programming error which causes the intended operation to be performed incorrectly.

Byte - a string binary* digits (bits) treated as a unit, often representing one data character*. The computer's memory capacity is often expressed as the number of bytes available. For example, a computer with 16K bytes of memory has about 16,000 bytes available for storing programs and data.

Cartridges - preprogrammed ROM* modules which are easily inserted in the SEGA computer to extend its capabilities.

Character - a letter, number, punctuation symbol, or special graphics symbol.

Command - an instruction which the computer performs immediately. Commands are entered with no preceding line number.

Concatenation - linking two or more strings* to make a longer string. The "+" is the concatenation operator.

Constant - a specific numeric or string* value. A numeric constant is any real number such as 1.2 or -9054. A string constant is any combination of up to 248 characters enclosed in quotes, such as "HELLO THERE" or "275 FIRST ST."

Cursor - a symbol which indicates where the next character* will appear on the screen when you press a key.

Data - basic elements of information which are processed or produced by the computer.

Default - a standard character or value which the computer assumes if certain specifications are omitted within a statement* or a program*.

Device - (see Accessory Devices)

Disk - a mass storage device capable of random and sequential access.

Display - (noun) the video screen; (verb) to cause characters to appear on the screen.

Execute - to run a program; to perform the task specified by a statement* or command*.

Exponent - a number indicating the power to which a number or expression* is to be raised; usually written at the right and above the number. For example, $2^3 = 2x2x2$ $2^8 = 2x2x2x2x2x2x2x2$. In SEGA BASIC the exponent is entered following the letter "E" in scientific notation*. For example, $2^3 = 2$ 8; $1.3 \times 10^{25} = 1.3E25$.

Expression - a combination of constants, variables, and operators which can be evaluated to a single result. Included are numeric, string, and relational expressions.

File - a collection of related data records stored on a device; also used interchangeably with device* for input/output equipment which cannot use multiple files, such as a line printer.

Function - a feature which allows you to specify as "single" operations a variety of procedures, each of which actually

contains a number of steps; for example, a procedure to produce the square root via a simple reference name.

Graphics - visual constructions on the screen, such as graphs, patterns, and drawings, both stationary and animated. SEGA BASIC has build-in subprograms which provide easy to-use colour graphic capabilities.

Hardware - the various devices which comprise a computer system, including memory. the keyboard, the screen, disk drives, line printers, etc.

Hertz(HZ) - a unit of frequency. One Hertz = one cycle per second.

Hexadecimal - a base .16 number system using 16 symbols, 0.9 and A-F. It is used as a convenient "shorthand" way to express binary* code. For example, 1010 in binary = A in hexadecimal, 11111111 = FF. Hexadecimal is used in constructing patterns for graphics characters in the PATTERN subprogram.

Increment - a positive or negative value which consistently modifies a variable*.

Input - (noun) data* to be placed in computer memory; (verb) the process of transferring data into memory.

Input Line - the amount of data* which can be entered at one time. In SEGA BASIC, this is 248 characters.

Internal date-format - data* in the form used directly by the computer. Internal numeric data is 8 bytes* long plus 1 byte which specifies the length. The length for internal string data is one byte per character in the string* plus one length-byte.

Integer - a whole number, either positive, negative or zero.

I/O - Input/Output; usually refers to a device function. I/O is used for communication between the computer and other devices (e.g. keyboard, disk)

Iteration - the technique of repeating a group of program statements; one repetition of such a group. See Loop.

Line - see input line, print line, or program line.

Loop - a group of consecutive program lines which are repeatedly performed, usually a specified number of times.

Mantissa - the base number portion of a number expressed in scientific notation*. In 3.264E + 4, the mantissa is 3.264.

Mass Storage Device - an accessory device*, such as a cassette recorder or disk drive, which stores programs and/or data* for later use by the computer. This information is usually recorded in a format readable by the computer, not people.

Memory - see RAM, and ROM, and mass storage device.

Noise - various sounds which can be used to produce interesting sound effects. A noise, rather than a tone, is generated by the SOUND subprogram* when commands 4 and 5 are specified.

Null String - a string* which contains no characters and has zero length.

Number Mode - the mode assumed by the computer when it is automatically generating program line* numbers for entering or changing statements.

Operator - a symbol used in calculations (numeric operators) or in relationship comparisons (relational operators). The numeric operators are + ,-, * ,/ ,. The relational operators are , , = , = , ., = , = , =

Overflow - the condition which occurs when a rounded value greater than 9.9999999999999E + 99 or less than -9.9999999999999E-99 is entered or computed. When this happens, a warning is displayed, and the program* stops.

Output - (noun) information supplied by the computer; (verb) the process of transferring information from the computer's memory onto a device, such as a screen, line printer, or mass storage device*.

Parameter - any of a set of values that determine or affect the output of a statement* or function*.

Print Line - a 38-position line used by the PRINT statement.

Program - a set of statements which tell the computer how to perform a complete task.

Program Line - a line containing a single statement*. The maximum length of a program line is 248 characters*.

Pseudo-random number - a number produced by a definite set of calculations (algorithm) but which is sufficiently random to be considered as such for some particular purpose. A true random number is obtained entirely by chance.

Software - various programs which are executed by the computer, including programs built into the computer. Cartridges* programs, and programs entered by the user.

Statement - an instruction preceded by a line number in a program. In SEGA BASIC, more than one statement is allowed in a program line*.

RAM - random access memory; the main memory where program statements and data* are temporarily stored during program execution*. New programs and data can be read in, accessed, and changed in RAM. Data stored in RAM is erased whenever the power is turned off of BASIC is exited.

Reserved Word - in programming languages, a special work with a predefined meaning. A reserved word lmust be spelled correctly, appear in the proper order in a statement* or command*, and cannot be used as a variable* name.

ROM - read-only memory; certain instructions for the computer are permanently stored in ROM and can be accessed but cannot be changed. Turning the power off does not erase ROM.

Run Mode - when the computer is executing* a program, it is in Run Mode. Run Mode is terminated when program execution ends normally or abnormally. You can cause the computer to leave Run Mode by pressing BREAK during program execution (see Breakpoint*).

Scientific Notation - a method of expressing very large or very small numbers by using a base number (mantissa*) times ten raised to some power (exponent*). To represent scientific notation in SEGA BASIC enter the sign, then the mantissa, the letter E, and the power of ten (preceded by a minus sign if negative). For example, 3,264; -2.47E -17.

Scroll - to move the text on the screen so that additional information can be displayed.

String - a series of letters, numbers and symbols treated as a unit.

Subprogram - a predefined general-purpose procedure accessible to the user through the statement in SEGA BASIC. Subprograms extend the capability of BASIC and cannot be easily programmed in BASIC.

Subroutine - a program segment which can be used more than once during the execution* of a program, such as a complex set of calculations of a print routine. In SEGA BASIC a subroutine is entered by a GOSUB statement and ends with a RETURN statement.

Subscript - a numeric expression which specifies a particular item in an array*. In SEGA BASIC the subscript is written in parentheses immediately following the array name.

Underflow - the condition which occurs when the computer generates a numeric value greater than -1E-100, less than 1E-100, and not zero. When an underflow occurs, the value is replaced by zero.

Variable - a name is given to a value which may vary during program execution. You can think of a variable as a memory location where values can be replaced by new values during program execution.

***See definition in Glossary**